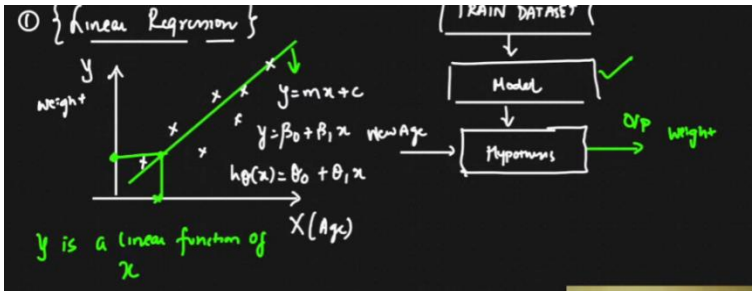
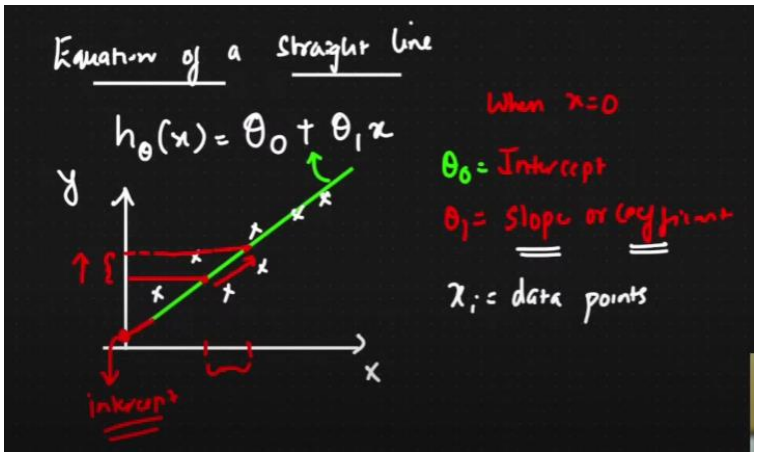
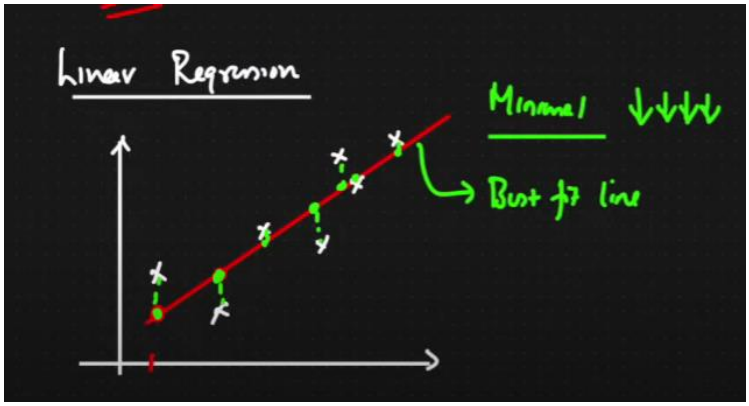
1) Linear Regression:





Θ0: at what point you are meeting the y-axis (intercept)

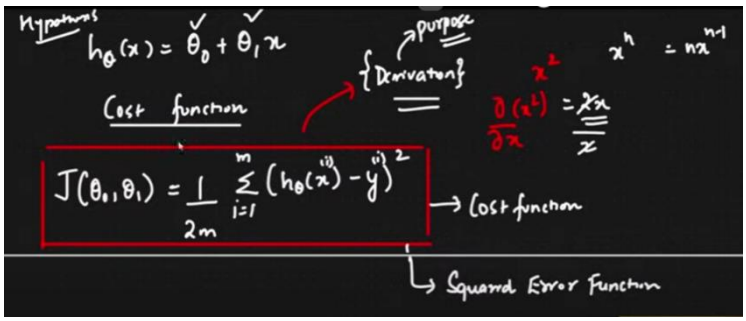
Θ1: unit movement of x & y axis



Best fit line: the distance between the predicted & the real points should be minimum!

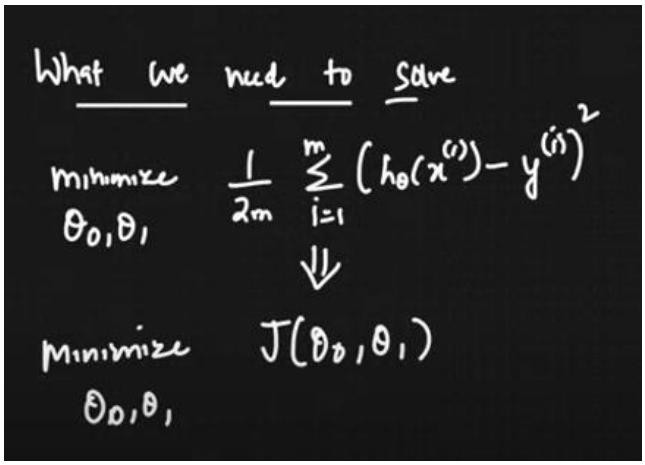
Will get it using, cost function also called as a distance formula and a squared error function!

green dot: real pts. & white circle: predicted pts.



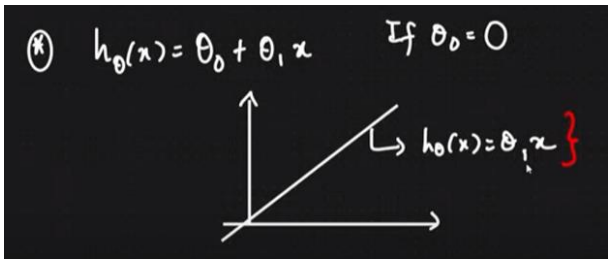
1) precited pts – real pts: **hΘ(x) - y**
2) **squaring**: to get rid of -ve values
3) **∑:** summation **i = 1** to complete **m**, m: no. of data pts, distance between predicted & real pts
4) **1/m**: average, **1/2m**: derivation purpose, equation simpler

Later, will minimize the cost function (**Θ0, Θ1)** as it's also called Squared Error Function!



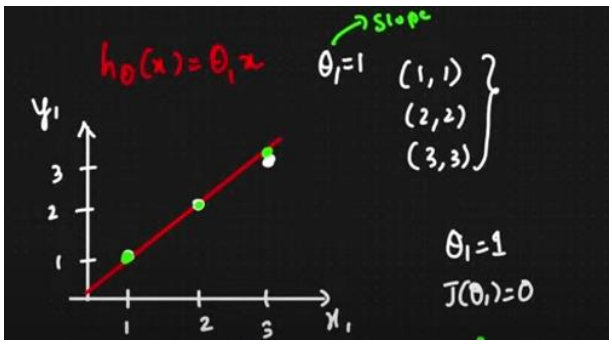
2) Comparing w.r.t hypothesis testing & cost function with an example:

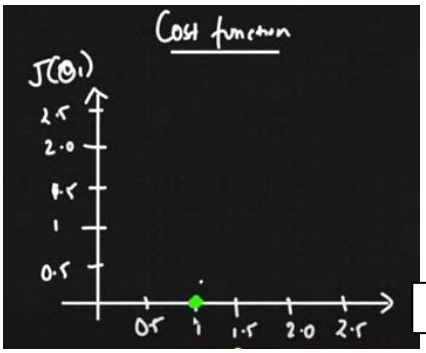
when, Θ0 = 0:



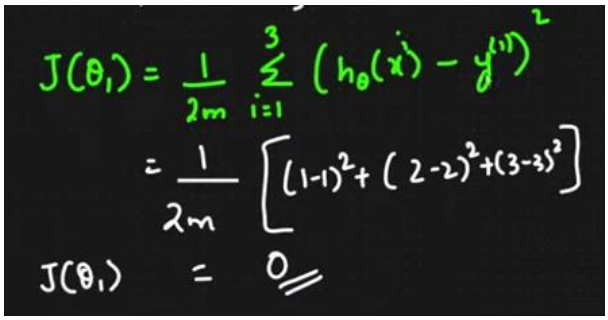
i) when, Θ1 = 1: (Θ0 passing through origin) (green circle: real pts.; white circle: predicted pts.)

hypothesis testing:



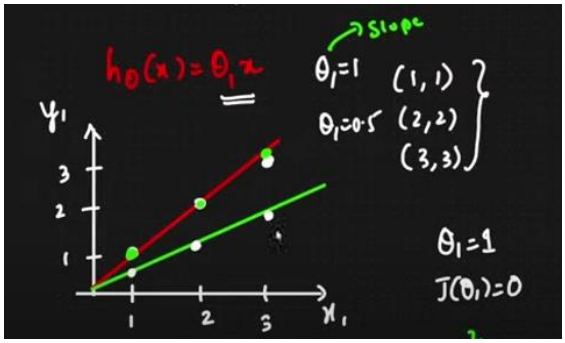


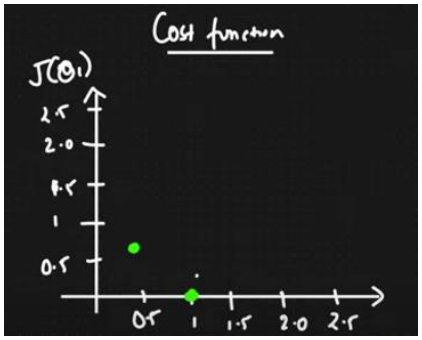
cost function:



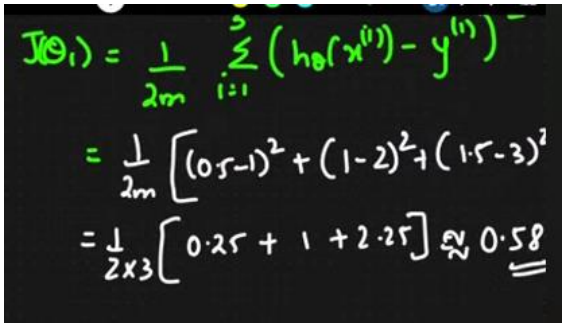
ii) when, Θ1 = 0.5:

hypothesis testing: (green line), (green circle: real pts.; white circle: predicted pts.)

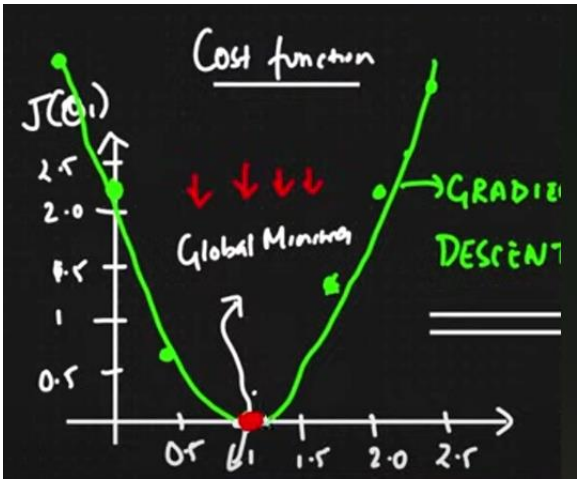




cost function:



3) Gradient Descent:



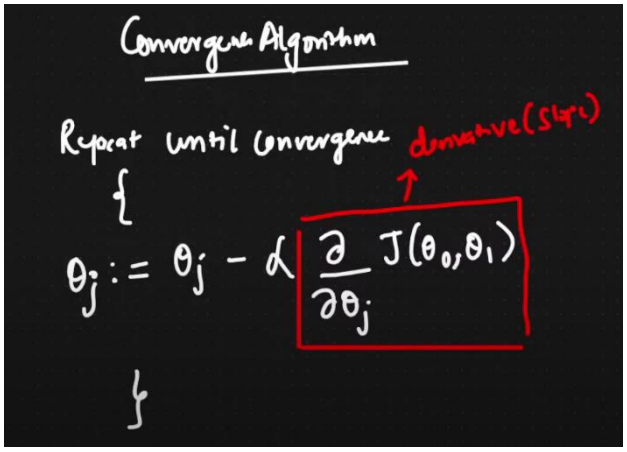
"GRADIENT DESCENT" helps to make sure that we get the right Θ1 value!

The most suitable point is the point which has the minimum distance between the real pt and the predicted pt., also known as "GLOBAL MINIMA"

Here, it's 1!

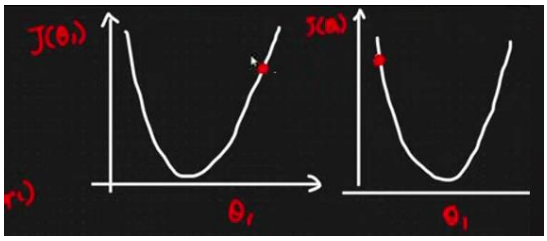
Here, we are assuming Θ1 value and moving ahead but this isn't the right way! The right way should be to find the global minima in the least attempts; so, for this will use something called as: "CONVERGENCE ALGORITHM".

4) Convergence Algorithm: Repeat the equation until it's convergence!



α = learning rate

1) find the slope:

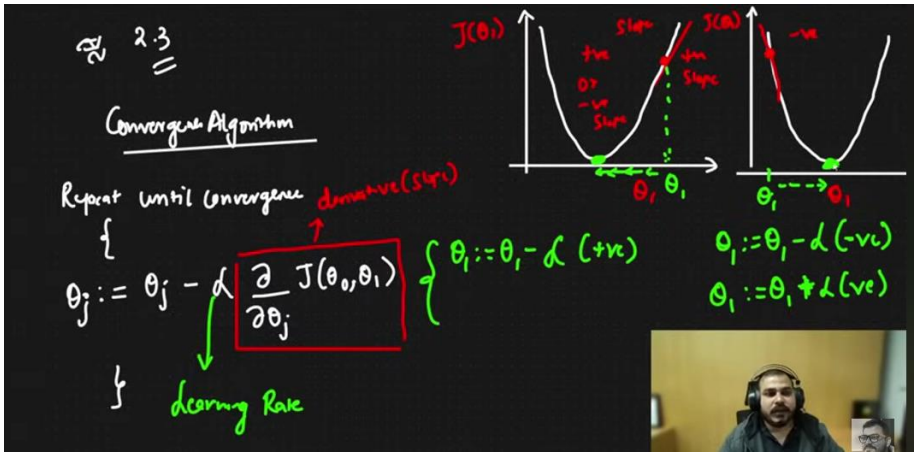


+ve flow -ve flow

i) initial pts marked with red!

ii) apply the derivative on the red pt to find the slope (right: +ve, left: -ve)

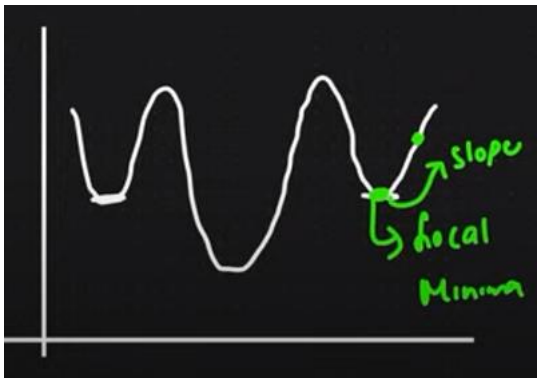
iii) apply convergence algorithm to get the global minima

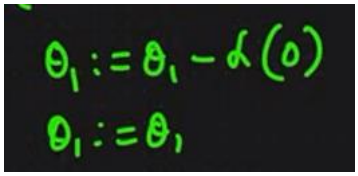


iv) what is α a learning rate?

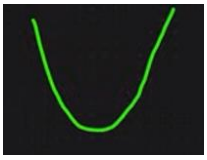
- ➔ Due to the speed the pt reaches from the initial to the global minima!
- ➔ It shouldn't be bigger nor smaller!
- ➔ So, it's good to keep α = 0.01

5) What if the cost function has the local minima?



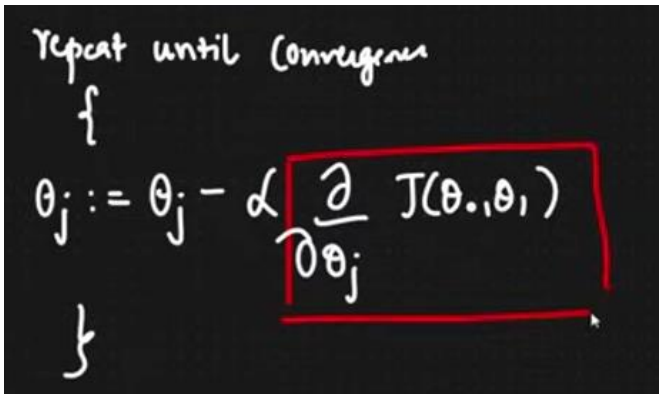


i) usually, in gradient decent and this type of equations (cost function) we don't find any issue, as we get a:

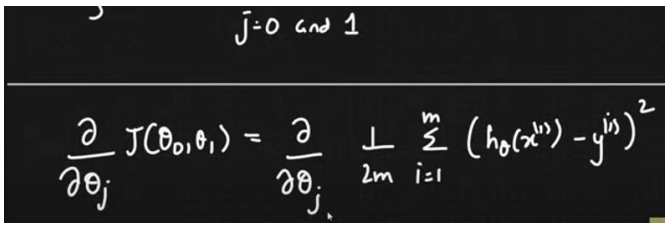


ii) but in the deep learning when we are learning about gradient descent and ann, we get lots of local minima! Because of that we have various gradient descent algorithms, like: rms prop, adam optimizes which will solve the specific problem!

6) Gradient Descent Algorithm: (MOST IMPORTANT, USES ON DAILY BASIS)



Find:



If j = 0:

$$j=0 \Rightarrow \frac{\partial}{\partial \theta_0} J(\theta_0,\theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

1/m instead of 1/2m just because when will find derivative of:

$\frac{1}{2m} x^2$ will get → $\frac{x}{m}$

And so, we have written: $\frac{1}{m}$

$$j=0 \Rightarrow \frac{\partial}{\partial \theta} J(\theta_0,\theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x - y^{(i)}\right)^2$$

Here, we have written the hypothesis testing formula, in the place of hΘ(x)!

Similarly:

$$\begin{cases} j=0 \Rightarrow \frac{\partial}{\partial \theta_0} J(\theta_0,\theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \\ j=1 \Rightarrow \frac{\partial}{\partial \theta_1} J(\theta_0,\theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)} \end{cases} \quad h_\theta(x) = \theta_0 + \theta_1 x$$

$x^2 \Downarrow 2x$

Here, j=1 has x^i in the end; as we have found the derivative of hΘ(x) from j=1, removing the square from j=1, similarly did for j=0, in j=0 there's no x^i as its 0!

Finally, the convergence algorithms are:

Repeat until convergence {

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

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

}

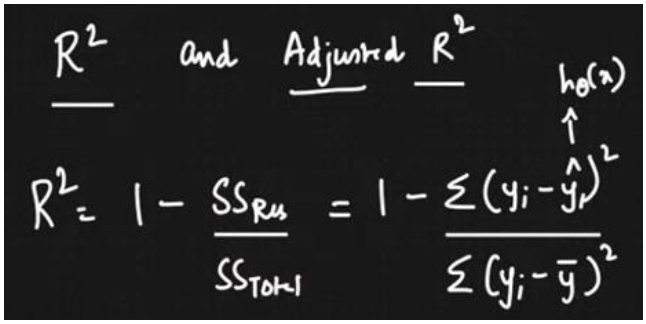
Will have lots of convex functions when we have multiple features: x1, x2, x3, …, xn

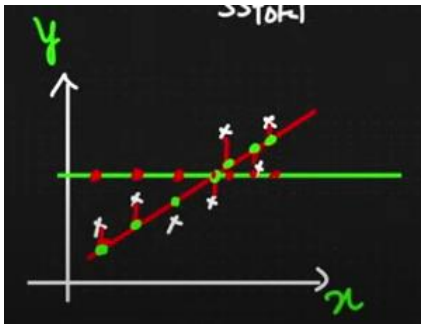
Here, gradient descent will look like a 3D curve: 

There will be lots of slopes & our aim will be to get the global minima, down the line!

7) Performance Matrix: a) R^2 & b) Adjusted R^2

a) R^2 used to verify, how good our model is w.r.t linear regression!



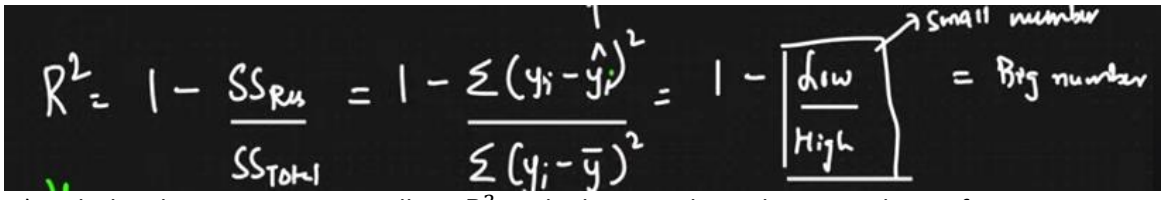


i) green dots: predicted, green line: mean

ii) Which one will be high?

This: $\Sigma(y_i-\hat{y}_i)^2$ or this: $\Sigma(y_i-\bar{y})^2$

second one; mean of a particular distance will always be higher!



iii) with the above given steps, will get R^2 as the bigger value, when we subtract form 1

iv) as much bigger the R^2, better the model!

v) can we get R^2 as a -ve number?

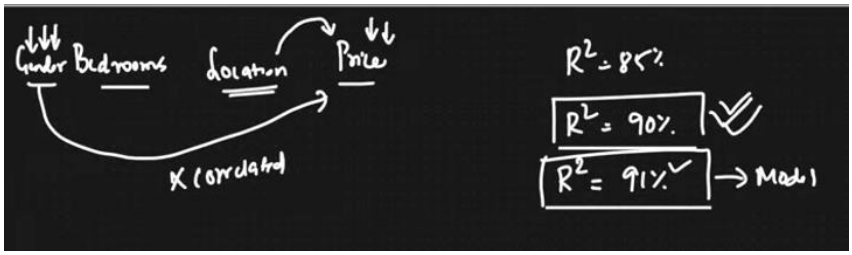
→ it's only possible when the best fit line is above the mean, due to which we get:

$1-\frac{\Sigma(y_i-\hat{y}_i)^2}{\Sigma(y_i-\bar{y})^2}$ = $1-\frac{High}{Low}$

usually, it doesn't happen; at least will try to fit a line which is good, having low distance between predicted and real pts then the mean.

vi) to get a good model and R^2 value, sometimes we take a column which isn't necessary / not correlated to the target variable!

Example:



To get rid of this situation, will use Adjusted R^2

b) Adjusted R^2:

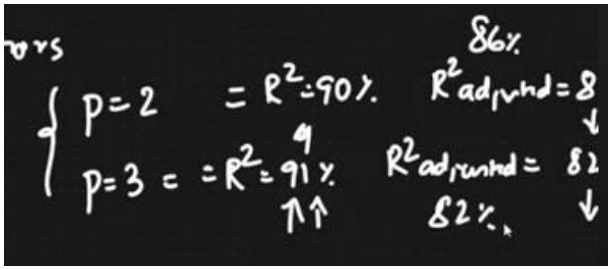
$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where

R^2 Sample R-Squared

N Total Sample Size

p Number of independent variable



Why have the Adjusted R^2 decrease?

When the independent variables are strongly related to the dependent variable, both R^2 and Adjusted R^2 will go up.

However, if an independent variable doesn't really help explain the dependent variable, R^2 might still be high, even though the model isn't that useful.

In this case, Adjusted R^2 will be lower, giving a more accurate picture of how well the model works.

So, Adjusted R^2 helps fix the problem of R^2 being misleading, especially when there are many independent variables, some of which aren't helpful. It's a better way to choose the best model in these cases.

8) Overfitting & Underfitting:

Goal: Using cost function, get the minimum distance and find the best fit lines!

when, Θ0 = 0, J (Θ0, Θ1) = 0, we got the best fit line by training the sample data!

But this is not the case in the real-world scenario, we either get good result on the training dataset or on testing dataset; it's impossible to be exactly accurate on the training and testing dataset!

It happens due to overfitting & the underfitting, explained below:

Overfitting vs Underfitting:

Aspect	Underfitting	Overfitting
What it means	Model is too simple to capture important patterns.	Model is too complex and captures unnecessary details.
Training Performance	Poor (doesn't learn the patterns well).	Excellent (learns the training data too well).
Test Performance	Poor (doesn't generalize well to new data).	Poor (fails to generalize to new, unseen data).
Bias	High (makes strong assumptions and misses patterns).	Low (fits the training data very closely, even the noise).
Variance	Low (model doesn't change much across different data).	High (model's performance is sensitive to the training data).
Example	Child says "all fruit is the same size" and can't tell apples from oranges.	Child memorizes every small detail (like blemishes) and gets confused with new fruit.
Key Issue	Too simple, ignores important details.	Too complex, memorizes irrelevant details.
Ideal Solution	Make the model more complex, include important factors.	Simplify the model, avoid overcomplicating with unnecessary details.

To handle the overfitting and underfitting, will work on regularization (ridge & lasso)!

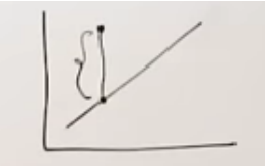
9) **Regularization**:

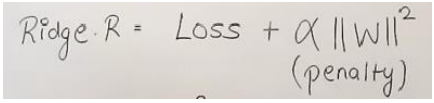
i) used to reduce the complexity of the model (mainly the training phase)

ii) shrink the magnitude value of a regression coefficient, if high!

iii) make it computational (should be cheap & take no time)

a) **Ridge Regression** (L2 regularization):

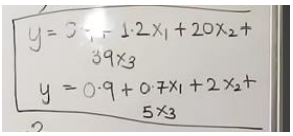




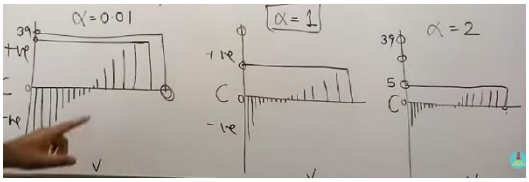
*loss = predicted value – actual value!
*to compensate the loss, we have added the penalty; due to this we reduced the loss by some amount & shrink the coefficient magnitude value

*α = constant (can be any value); *$\|w\|^2$: (vector of the coefficient) = $w_1^2 + w_2^2 + w_3^2 + ... + w_n^2$

Example:



→

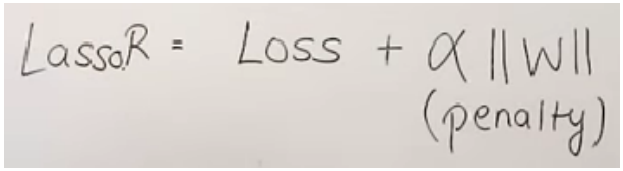


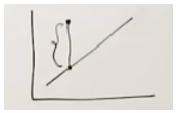
as, α value increases the coefficient of the magnitude decreases/shrinks/scale downs almost to 0;

x3 = 39 decreases to 5!

Achieved the regularization, by reducing the complexity & by reducing the expenses of the computational. Prevent overfitting!

b) **Lasso Regression** (L1 regularization):

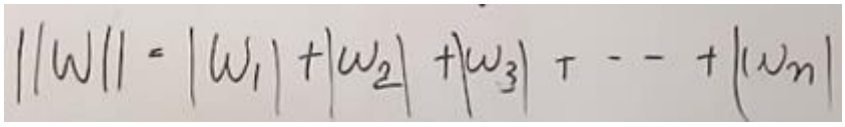




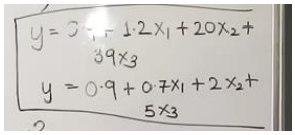
*loss = predicted value – actual value!

*to compensate the loss, we have added the penalty; due to this we reduced the loss by some amount & shrink the coefficient magnitude value

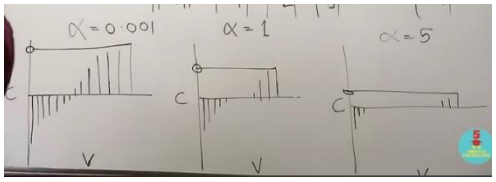
*α = constant (can be any value)



Example:



→



as, α value increases the coefficient of the magnitude decreases/shrinks/scale downs to exactly 0;

x2 = 20 decreases to 0!

Achieved the regularization, by reducing the complexity & by reducing the expenses of the computational.

Also, those who gets 0, those variables/features are not important and can be directly eliminated w.r.t variable y (a dependent variable), known as **feature selection**.

Prevent Overfitting & achieves Feature Selection!

- Try both the regularization and the one with good performance matrix, select/use that!

10) Assumption of Linear Regression:

i) if our features follow: Normal / Gaussian Distribution → model will get trained well!

if it doesn't follow: Normal / Gaussian Distribution, will try to do feature transformation to convert the data into Normal / Gaussian Distribution, if possible.

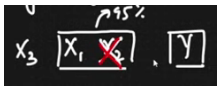
iii) Standardization: Scaling the data → z-score, $\mu=0, \sigma=1$

use whenever there's Gradient Descent and find the global minima value, as soon as possible!

Helps to optimize the model & use to increase the training time of the model.

iv) Linear regression works w.r.t linearity.

v) Check multicollinearity:



over here, if we try to see the collinearity of x1 & x2, and we get to know it's 95% correlated, will use just one among the 2 features & do the prediction!

Variation Inflation Factor (VIF) is also used to solve the multicollinearity.

There's also **Homoscedasticity**!

Reference:

1) Krish Naik - Linear Regression (from 33 minutes onwards)

2) 5 Minutes Engineering - Overfitting & Underfitting

3) 5 Minutes Engineering - Ridge Regression Explanation

4) 5 Minutes Engineering - Lasso Regression Explanation

5) Krish Naik - Assumption of Linear Regression (from 41:36 minutes onwards)