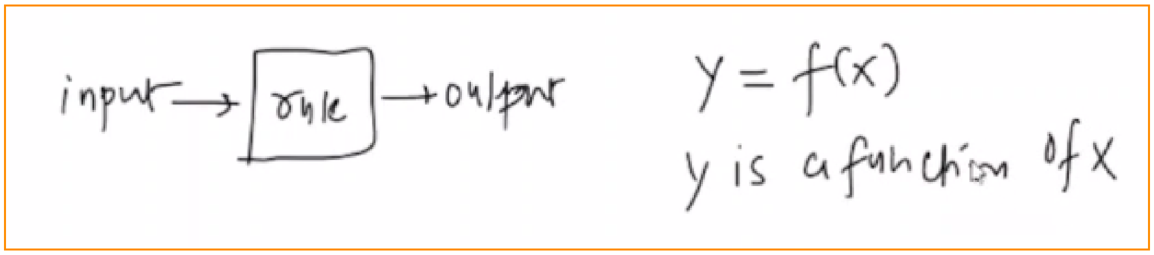
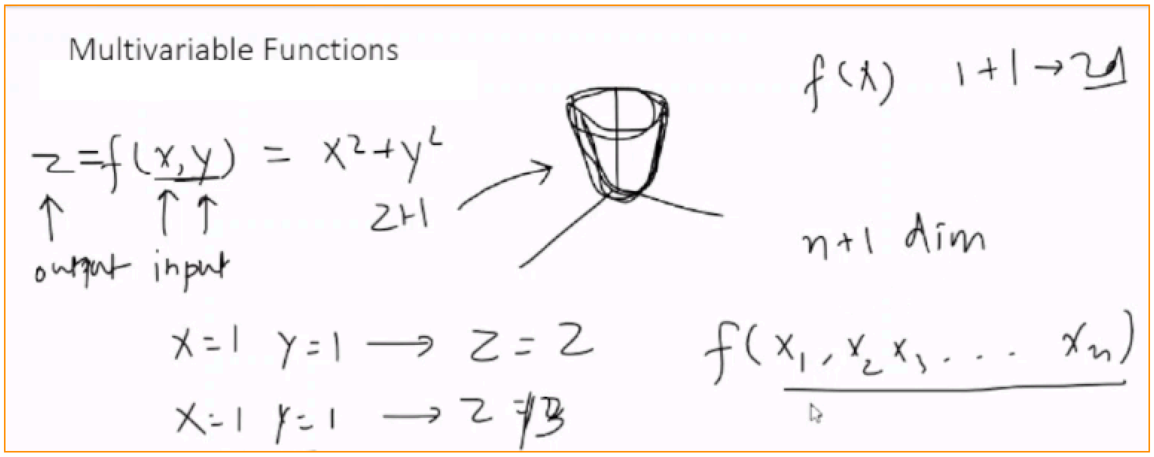
What are function?

- function is a mathetmatic rule which takes an input, do some processing based on mathematic rule & provides an output

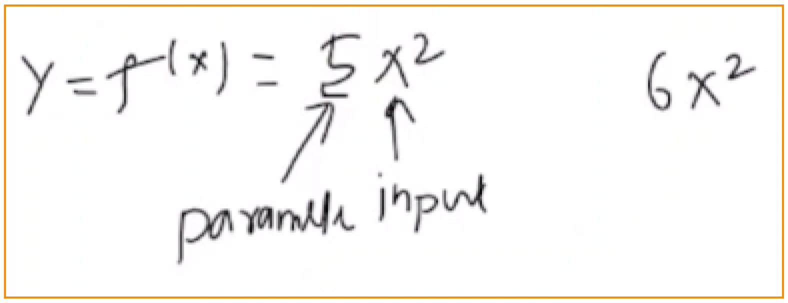


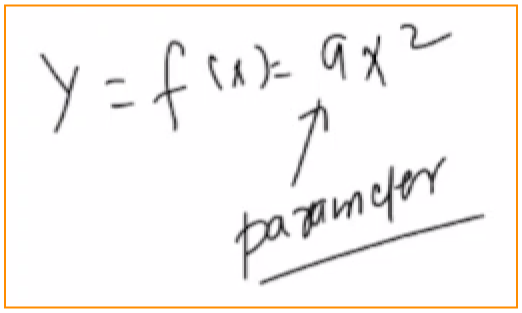
- x is input & y is output
- x is a rule like $x^2 + 2x$ so if we pass x = 2 then y == 8
- It can of any number of variables
- z = f(x,y) $(x^2 + Y^2)$
- a = f(x,y,z....n)
- dimension will be n + 1 where n is the count of input variable



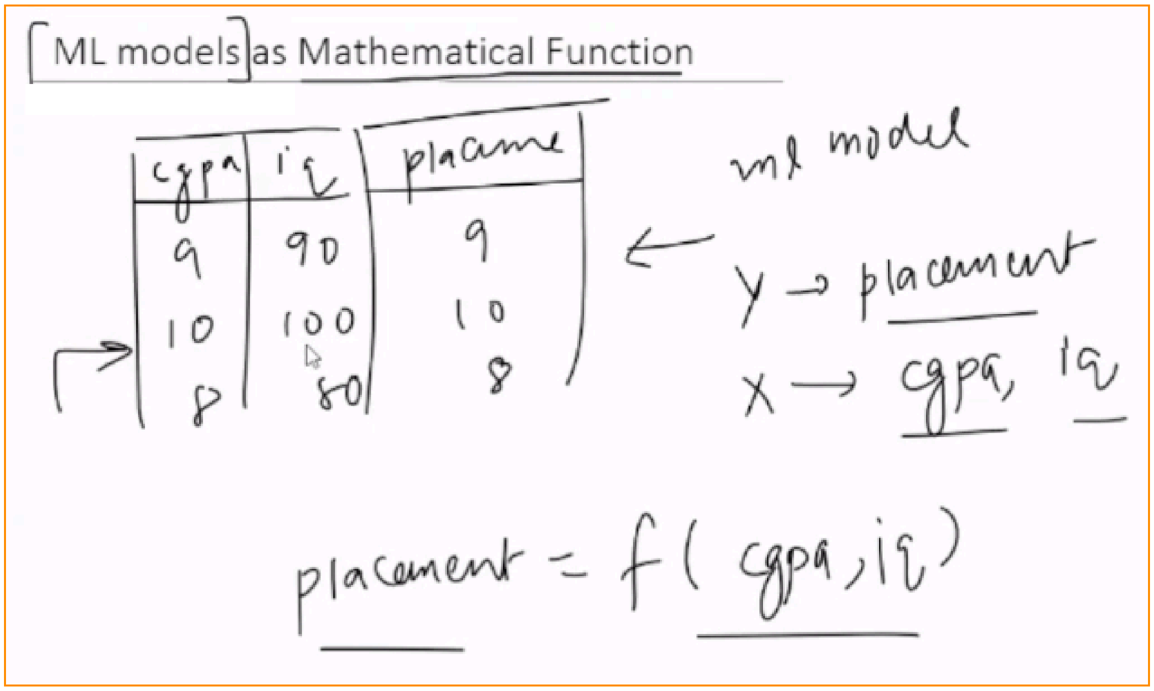
Parameters of function

- Parameters are the variables that influence the behaviour of function
- like in quadtratic function: $ax^2 + bx + c$. a,b & c are parameters
- $y = 5x^2$ & $y = 6x^2$ & x = 2
- like 5 ^ 6 will have different impact





ML Model are Mathematical Functions


ML models as Mathematical Function

cgpa	iq	placement
9	90	9
10	100	10
8	80	8

- There are 2 different types of ml models
    1. **Parametric:** data distribution assumption, fixed no of parameters irrespective of no of rows like linear regression
    2. **Non-parametric:** No assumptions like , no fixed parameters like decision tree

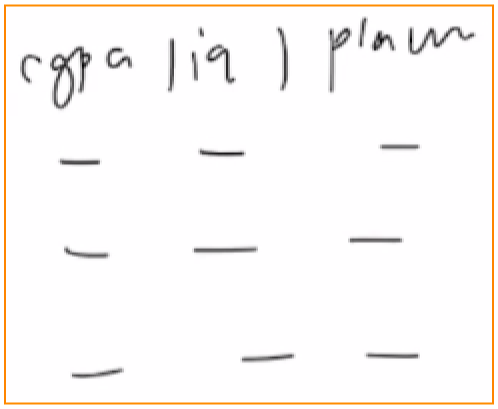
Lets understand the role of optimization

- lets assume we have below data of CGPA & IQ

- we have to build a prediction model to predict Placement amount
- we decided to use Linear Regression as that model
- so basically we're trying to build a function which takes 2 input
- output of that function is prediction amount ---> placement = f(cgpa,iq)
- since we're using linear regression so we assume that data is linear
- and the form will be: y = f(β0 + β1x1 + β2x2)
- and β0, β1, β2 will be parameters
- our goal after we've decided the algorithm is to find the most optimized values of β0, β1, β2 for the given data
- thats what we do using **Optimization**

How to perform Optimization?

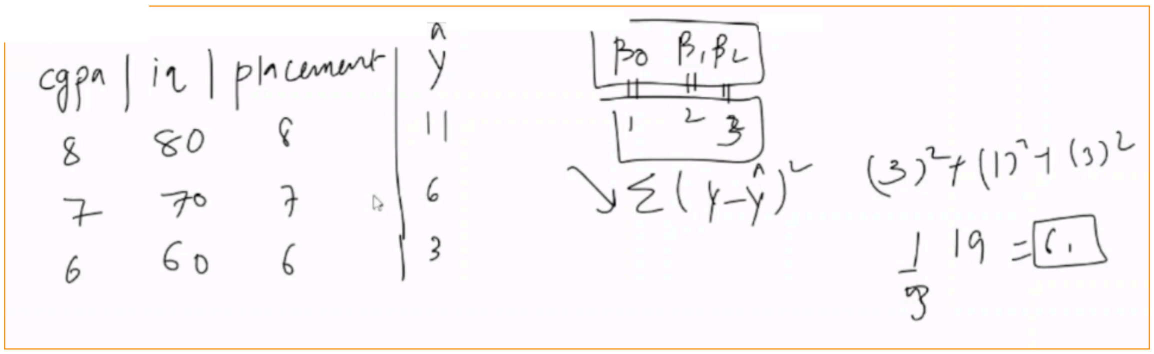
- Lets say we have a dataset & we need to find β0, β1, β2



cgpa	iq	placm
—	—	—
—	—	—
—	—	—

- we'll initialize with some random value, it could be anything like 1,1,1.. or 0.1,0.2,0.3 etc
- calculate predicted value ŷ (y-hat) using those random values
- now we have 2 columns as actual value y & predicted value as ŷ
- we'll take help of **Loss Function/cost function** which we have to decide MSE, MAE, RSME etc
- reason why we have multiple loss functions is problem type (regression or classification), presence of outliers, loss function interpretibility in same unit, diffrentiability, compatibility with ml model etc
- **Primary goal of training a ml model is to minimize the value of the loss function: That's Optimization**

How to calculate values of Parameters from Loss Function?



cgpa	iq	placement	$\hat{y}$
8	80	8	11
7	70	7	6
6	60	6	3

$\boxed{\beta_0 \ \beta_1 \beta_2}$

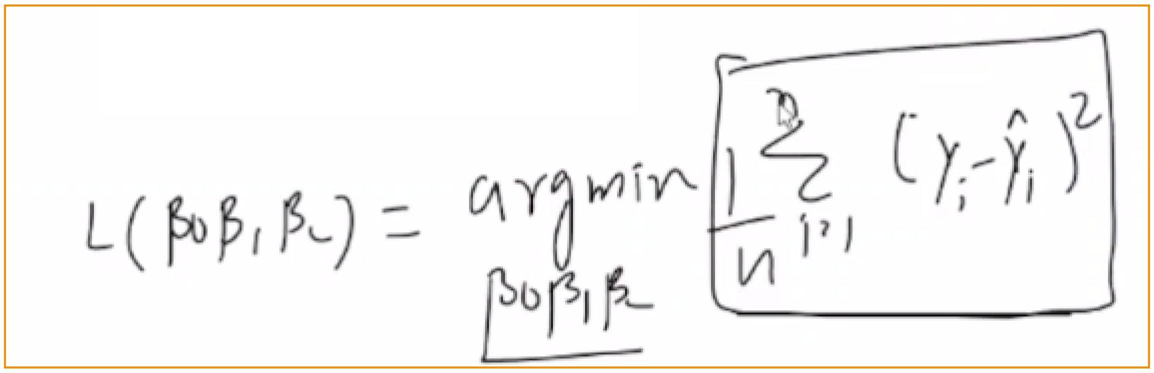
$\boxed{1 \quad 2 \quad 3}$

$\Sigma(y-\hat{y})^2$

$(3)^2+(1)^2+(3)^2$

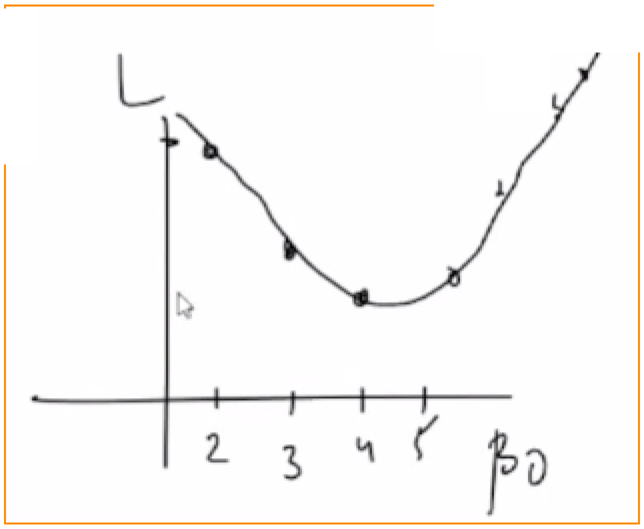
$\frac{1}{3}\ 19 = \boxed{6.}$

- Lets take a dummy data with random value & initiliza random beta values
- loss function is Σ(y - ŷ)2 == 6.6
- we have to find such values of β0, β1, β2 for which loss function is minimum



$$L(\beta_0\beta_1\beta_2) = \underset{\beta_0\beta_1\beta_2}{\operatorname{argmin}} \boxed{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$$

- we can provide random values & calculate loss function & plot it
- lets assume we already know the correct values of β1, β2.
- we've to find the value of β0
- we can provide random values & calculate loss function
- plot a graph which shows relationship b/w β0 & Loss function

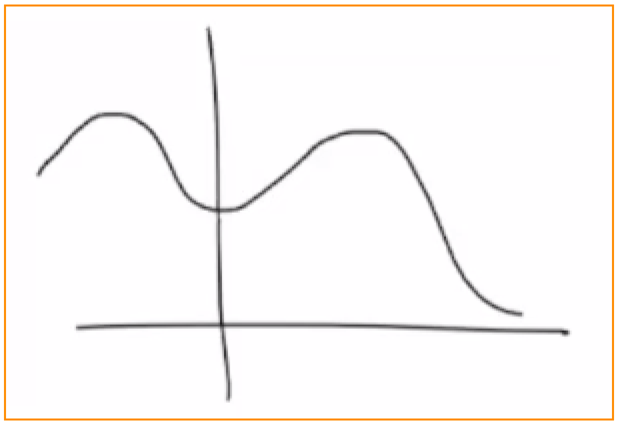


- graph will parabolic because of the squared term in loss function
- This is the point where we need to do optiization to arrive at the minimum point
- To locate the minimum value of y, the slope at that point must be == 0

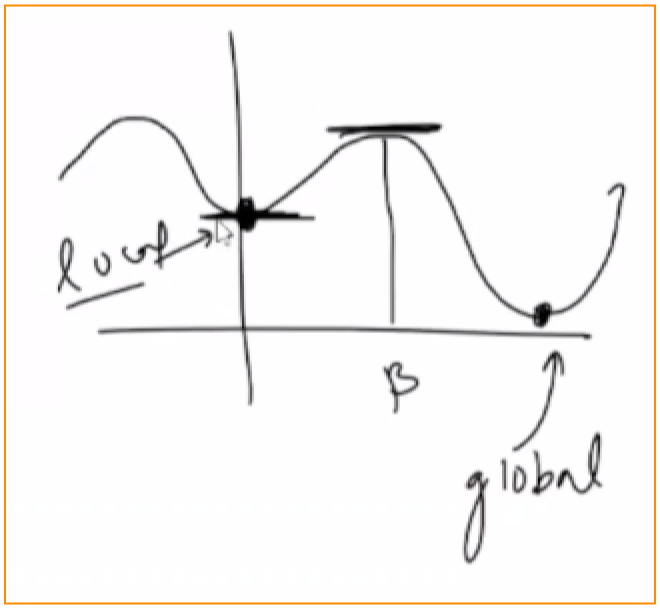
- So we have to do perform diffrentiation & set it equal to 0:

$$\frac{dL}{d\beta_0} = 0$$

- **BUT, Is this correct approach, Will it work all the time ??**
- It will certainly not.. This time our loss function was parabolic.. what if we have a different loss function whose graph looks like this:

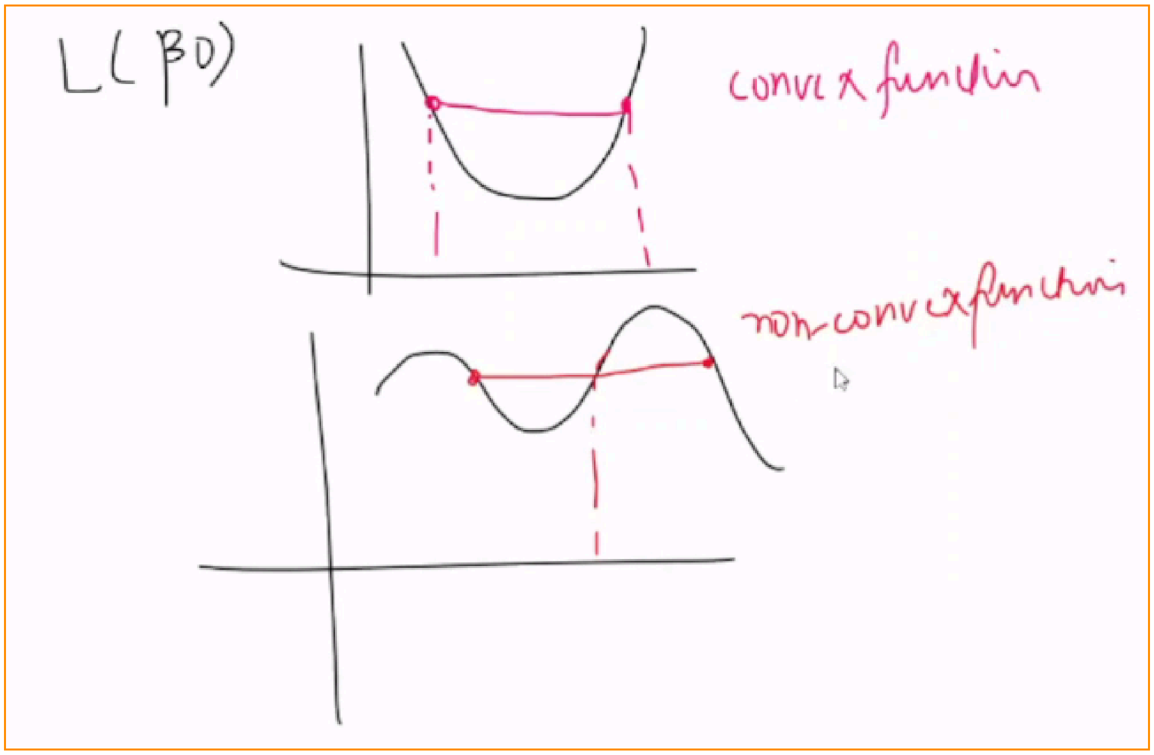


- There are 2 scenarios where slope == 0
  1. minima
  2. maxima
- There may be a chance instead of arriving at minimum value, we arrived at maximum value & β value is maxima
- Or, what if there are 2 minima
  1. local minima
  2. global minima
- we may stuck at local minima where as global minima is also available



- So finding a slope & setting it qual to 0 will give correct results in very limited scenarios
- **Problem with easy way**
    1. Non-Convexity: Loss function may not always be convex meaning it might have multiple local minima & maxima. It will fail in these scenarios
    2. Complexity: Some loss functions can be very complex & finding analytical solution could be computationally expensive or even impossible mostly in deep learning models
    3. Scalability: In large ml models with massive amount of data or features can be computationally very expensive
    4. Online Training/ Streaming data: When dealing with streaming data, retraining your model can be challenging. The key issue is that as you retrain the model with new data, the predictions are based on past information, while new data continuously arrives. This can lead to outdated predictions and suboptimal performance.

Convex & Non-Comvex loss functions



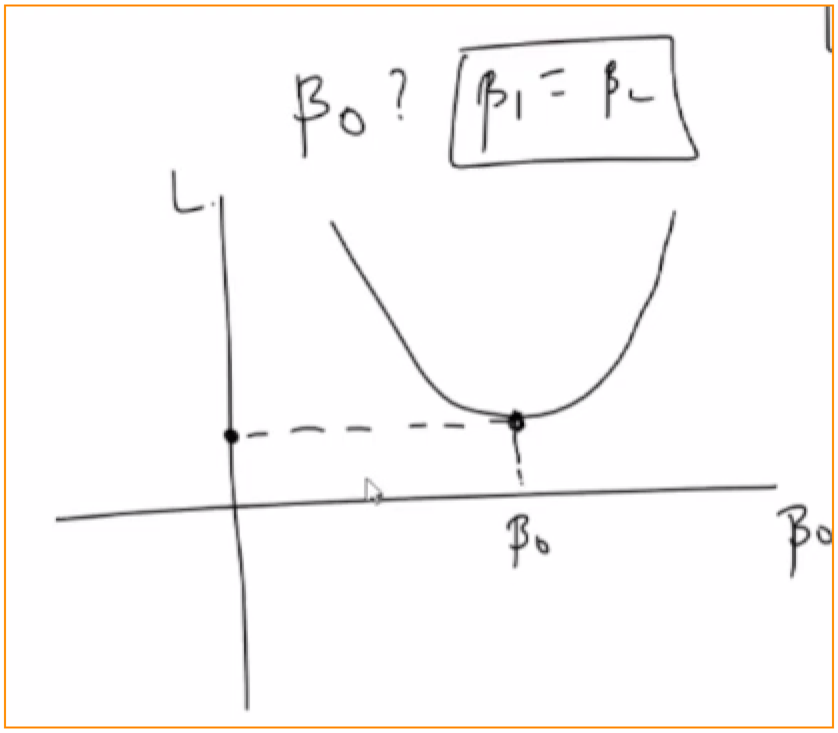
- If its a convex function, then it will have only 1 global minima but if its non-convex, there could be multiple minima & maxima

Gradient Descent

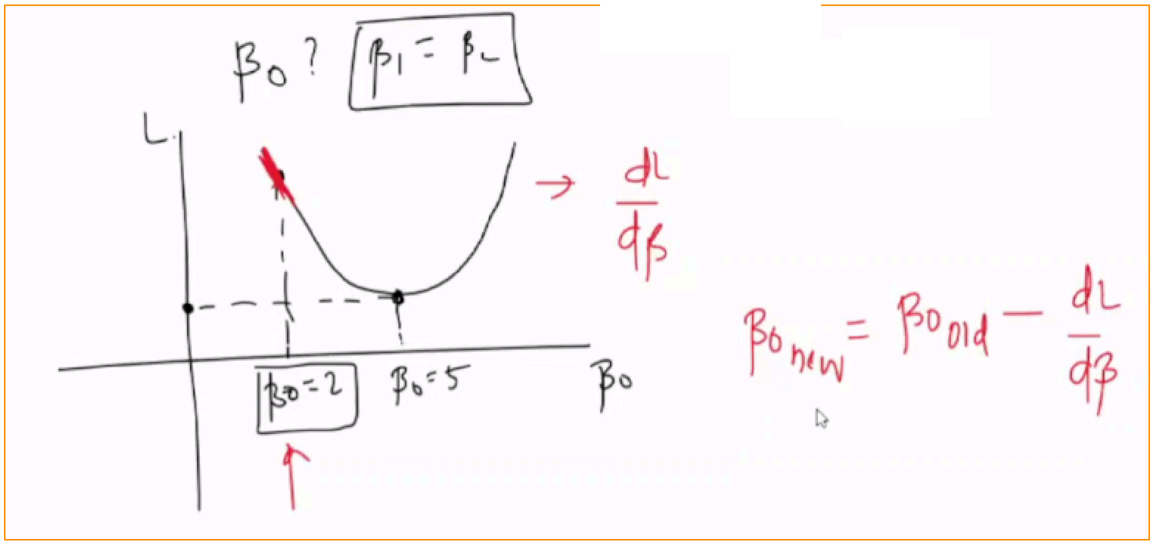
- In order to solve above problems we use multiple optimization technique
- **Gradient Descent** is most popular optimization technique that we use
- Newton method, Quadatric programming, Linear programming etc are some other optimiztion techniques
- we can use Gradient Descent in Linear Regression, Logistic regression, Deep Learning, Advance nueral networks
- In SVM & PCA, we use quaditric programming

How it works?

- Lets assume we already know the correct value of β1, β2
- we have to find β0
- so we initialize β0 with random value & calculate loss function
- and plot a grpah b/w loss function & β0



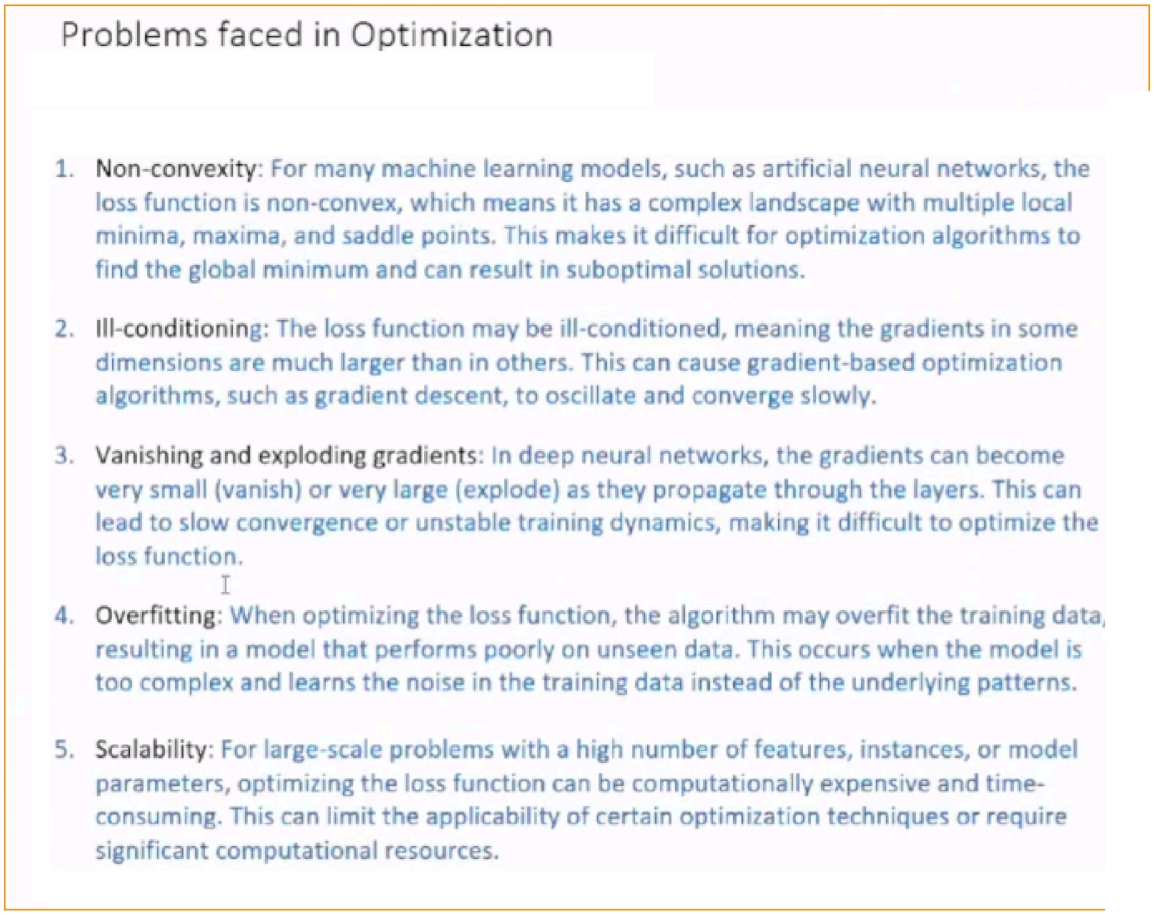
- and its a convex function
- we have to reach the lowest value of β0 as shown above for which loss function has lowest value
- lets start β0 as 2 & we have to arrive at 5, so we'll find slope & move towards opposite value of slope continoulsy
- eventually we reach at the lowest value
- there is something called "Learning Rate": which is the amount of which it jumps.. it cant be very high
- if its very high, jumpy will be very drastic & it may never reach minima
- generally keep it to 0.01



β_0 ? $\boxed{\beta_1 = \beta_2}$
L
$\rightarrow \frac{dL}{d\beta}$
$\boxed{\beta_0 = 2}$
$\beta_0 = 5$
β_0
$\beta_{0\,new} = \beta_{0\,old} - \frac{dL}{d\beta}$

$$\beta_{0_n} = \beta_{0_{old}} - \eta \frac{dL}{d\beta_0}$$

- In simple words... we calculate slope & move in opposite direction of where its increasing & eventually it reaches to a point where slop is 0 & beta new beta old will be same & thats the minima
- So atleast we wont reach maxima but local & global minima problem still persists
- To solve that, we can apply other its variants like Momentum, NAG, RMS prob, Adam etc
- we can use them to solve local minima problem
- In ML generally we wont have to use them since mostly loss function will be convex, we get to use them in Deep Learning



Problems faced in Optimization

1. Non-convexity: For many machine learning models, such as artificial neural networks, the loss function is non-convex, which means it has a complex landscape with multiple local minima, maxima, and saddle points. This makes it difficult for optimization algorithms to find the global minimum and can result in suboptimal solutions.
2. Ill-conditioning: The loss function may be ill-conditioned, meaning the gradients in some dimensions are much larger than in others. This can cause gradient-based optimization algorithms, such as gradient descent, to oscillate and converge slowly.
3. Vanishing and exploding gradients: In deep neural networks, the gradients can become very small (vanish) or very large (explode) as they propagate through the layers. This can lead to slow convergence or unstable training dynamics, making it difficult to optimize the loss function.
4. Overfitting: When optimizing the loss function, the algorithm may overfit the training data, resulting in a model that performs poorly on unseen data. This occurs when the model is too complex and learns the noise in the training data instead of the underlying patterns.
5. Scalability: For large-scale problems with a high number of features, instances, or model parameters, optimizing the loss function can be computationally expensive and time-consuming. This can limit the applicability of certain optimization techniques or require significant computational resources.

```
In [ ]:
```