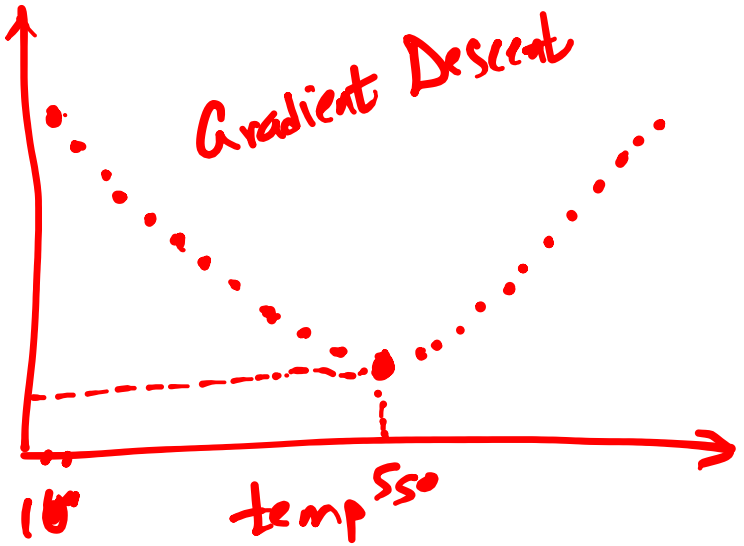
Optimization Techniques


Good Morning
Ice89


Good Morning
Ice89

Exhaustive Search

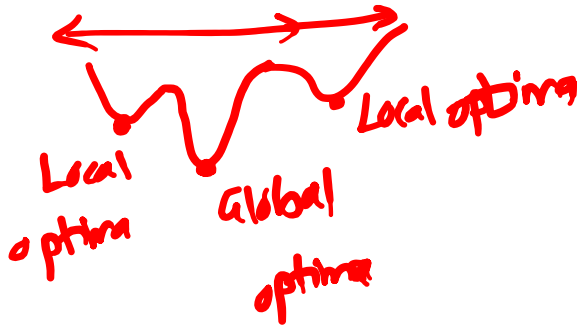
Suffering



10 °C
15
20
25
⋮
100 °C

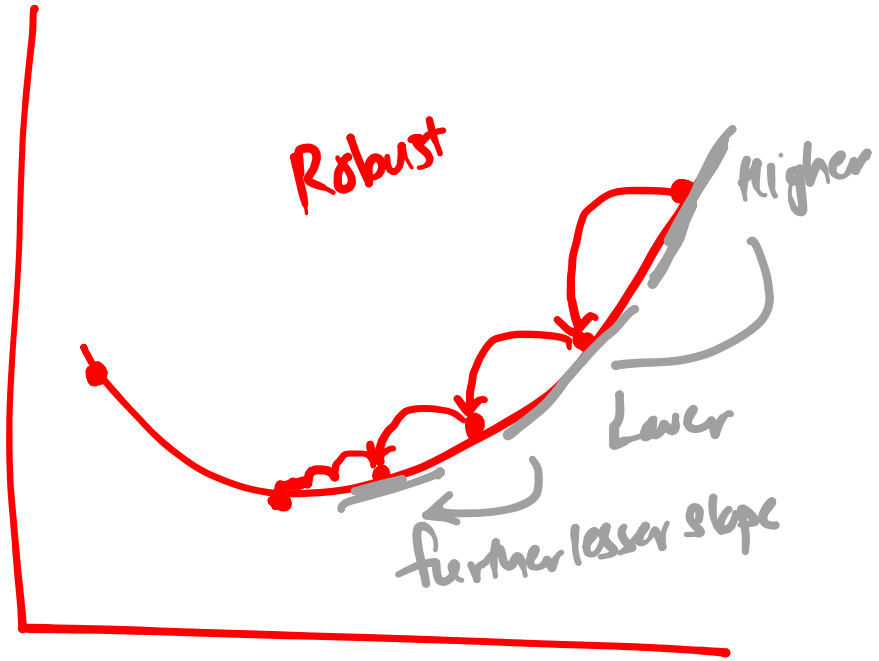
Best (Least Suffering)

Adv: Simple
Global optimum



Disadv: 1. Computationally expensive.

2. Get close to real min bcz of predefined Stepsize, but we never achieve minimum



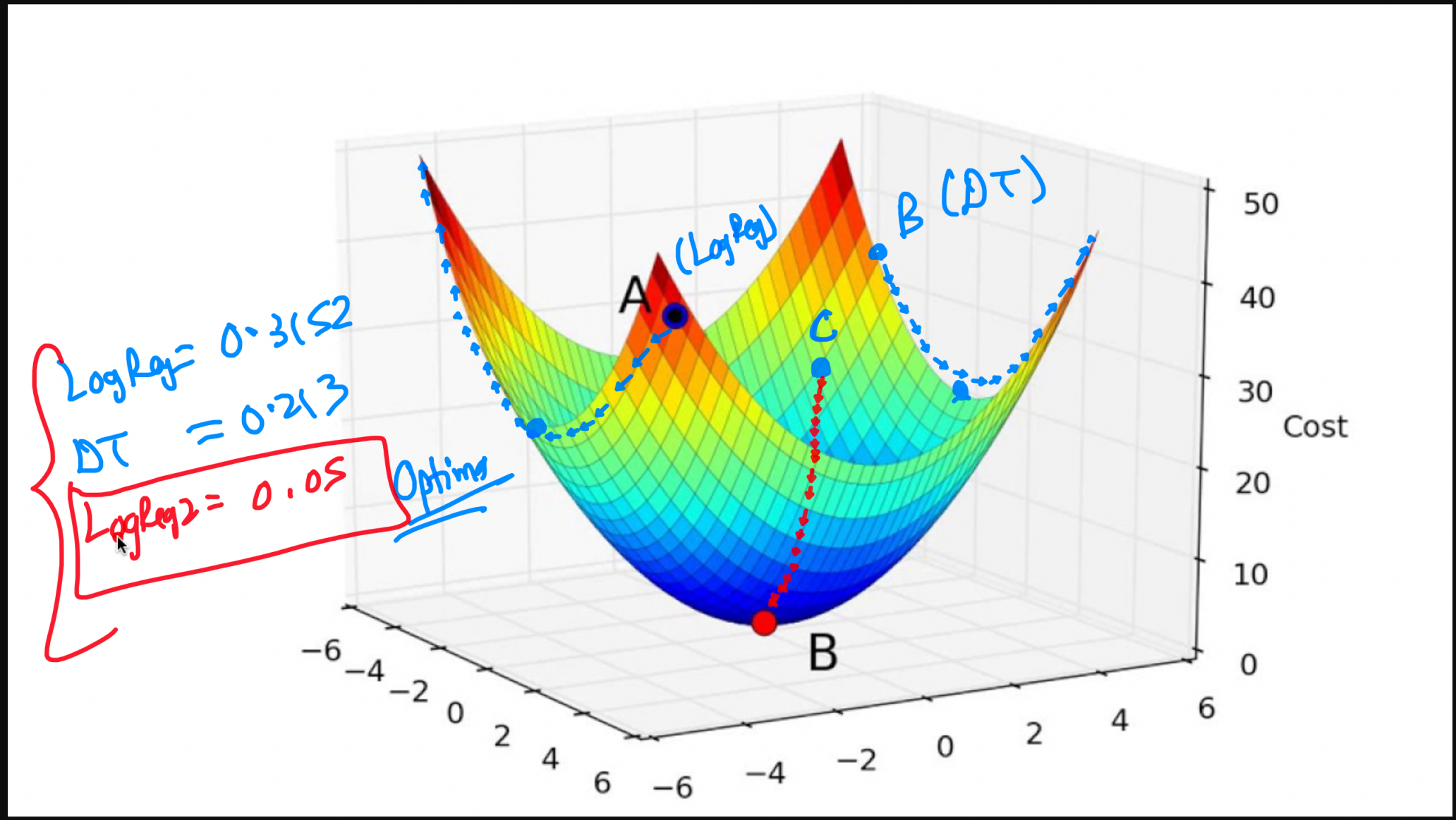

```
from sklearn.lin_mod imp. LogReg
obj=LogReg() # method='gradient'
obj.fit(X_train, y_train)
```

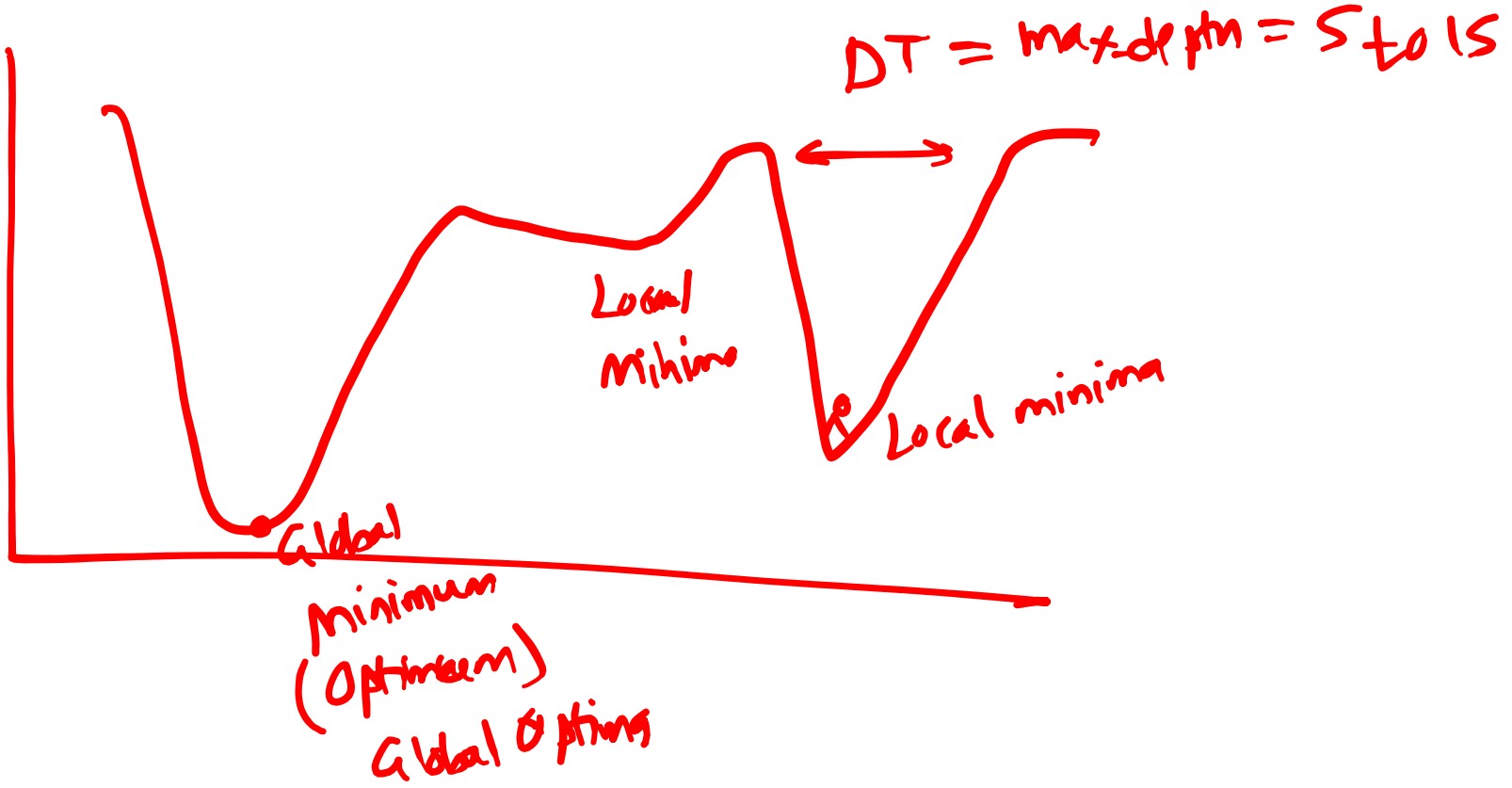
If slope is high, we jump higher

If slope is less, we jump closer.

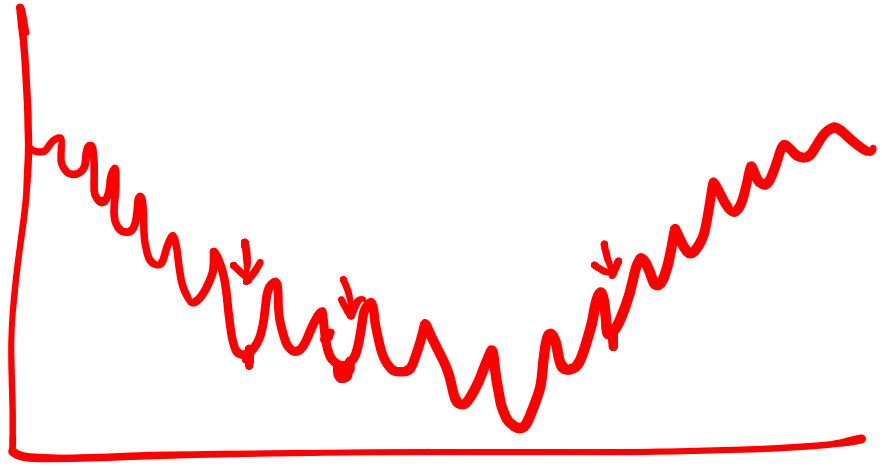
Step Size $\propto$ Slope

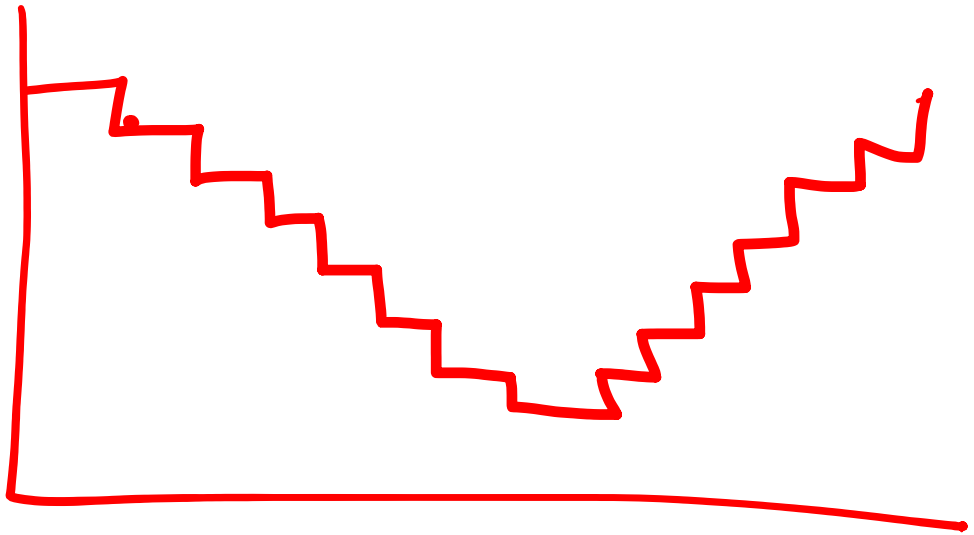
SS = Learning rate x Slope

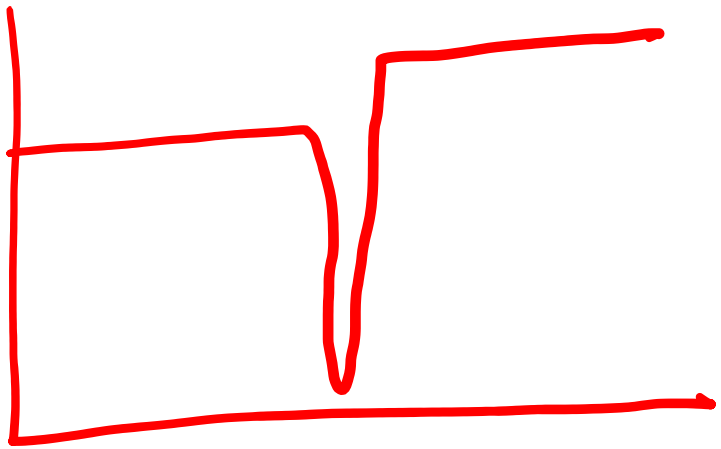

A
(LogReg)
B (DT)
C
B
LogReg = 0.352
DT = 0.213
LogReg2 = 0.05
Optimal
Cost
50
40
30
20
10
0
−6
−4
−2
0
2
4
6


DT = max depth = 5 to 15
Local Minima
Local minima
Global Minimum (Optimum)
Global Optima

Sometimes error function is not smooth. There are lot of places a marble could get stuck.





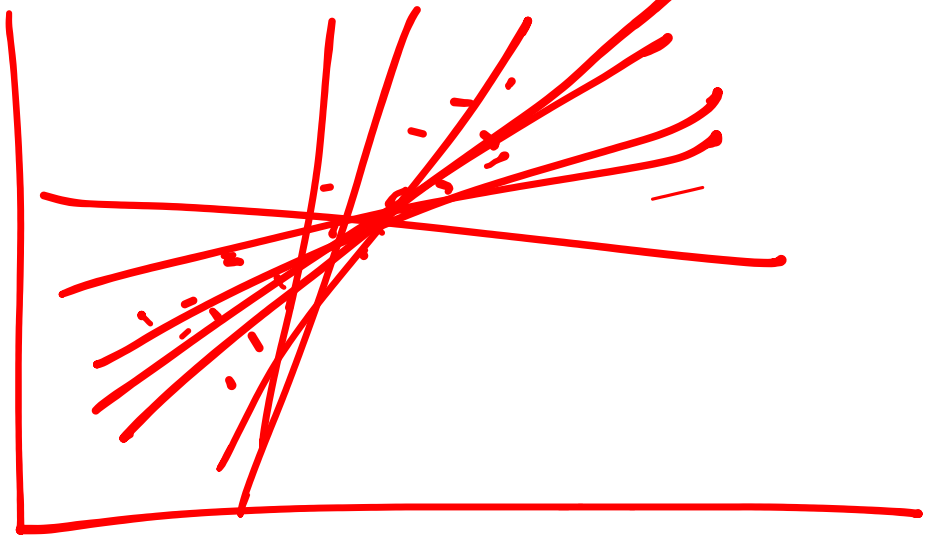


Cost Function : We look at the loss associated with each training example & then sum these values together for an overall cost of the entire dataset.

Batch Gradient Descent :

y	TV	Pred
	0	0 = error

LOBF



$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_1 - y_p)^2}$$

all n data points

assume n = 10 lakhs

e1
e2
e3
!
!
e25

least

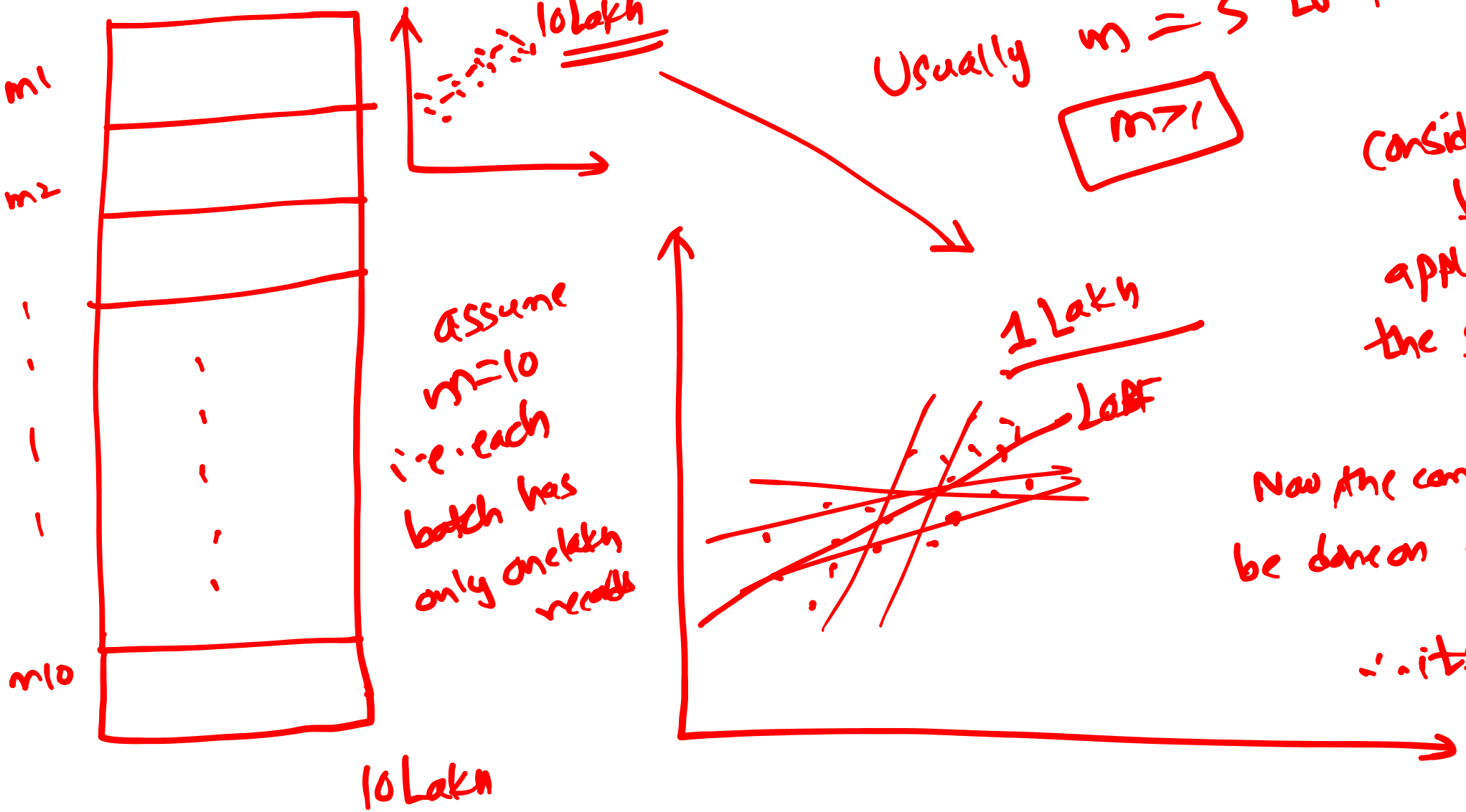
Disadv: Slow & Comp. expensive.

Mini – Batch Gradient Descent :

m mini batches

Usually m = 5 to 10

m>1

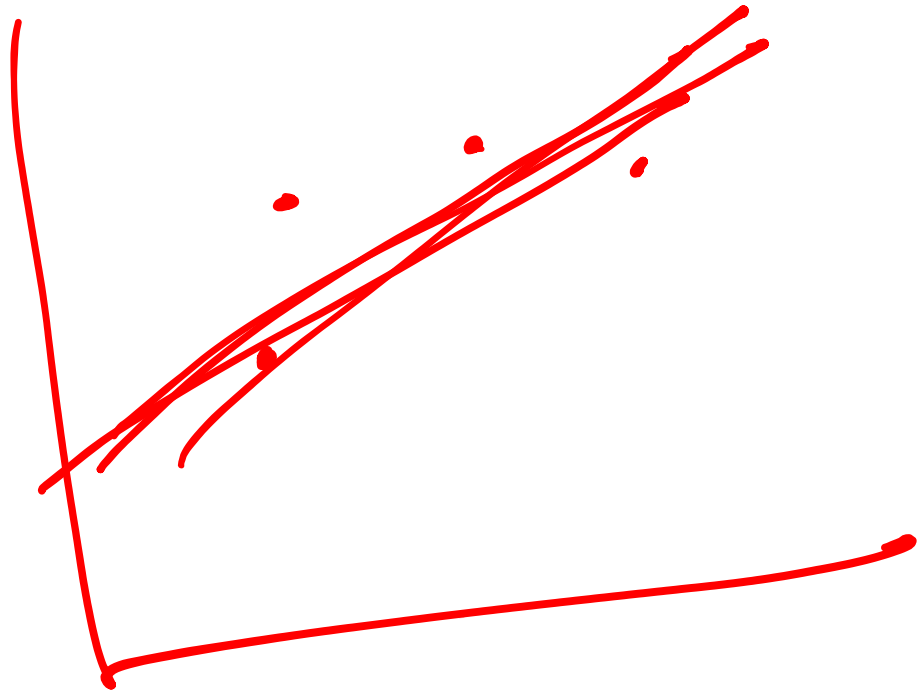


Consider any one batch & apply G.D. the same way.

Now the computation will be done on $\frac{1}{m}$ th dataset.

∴ its faster

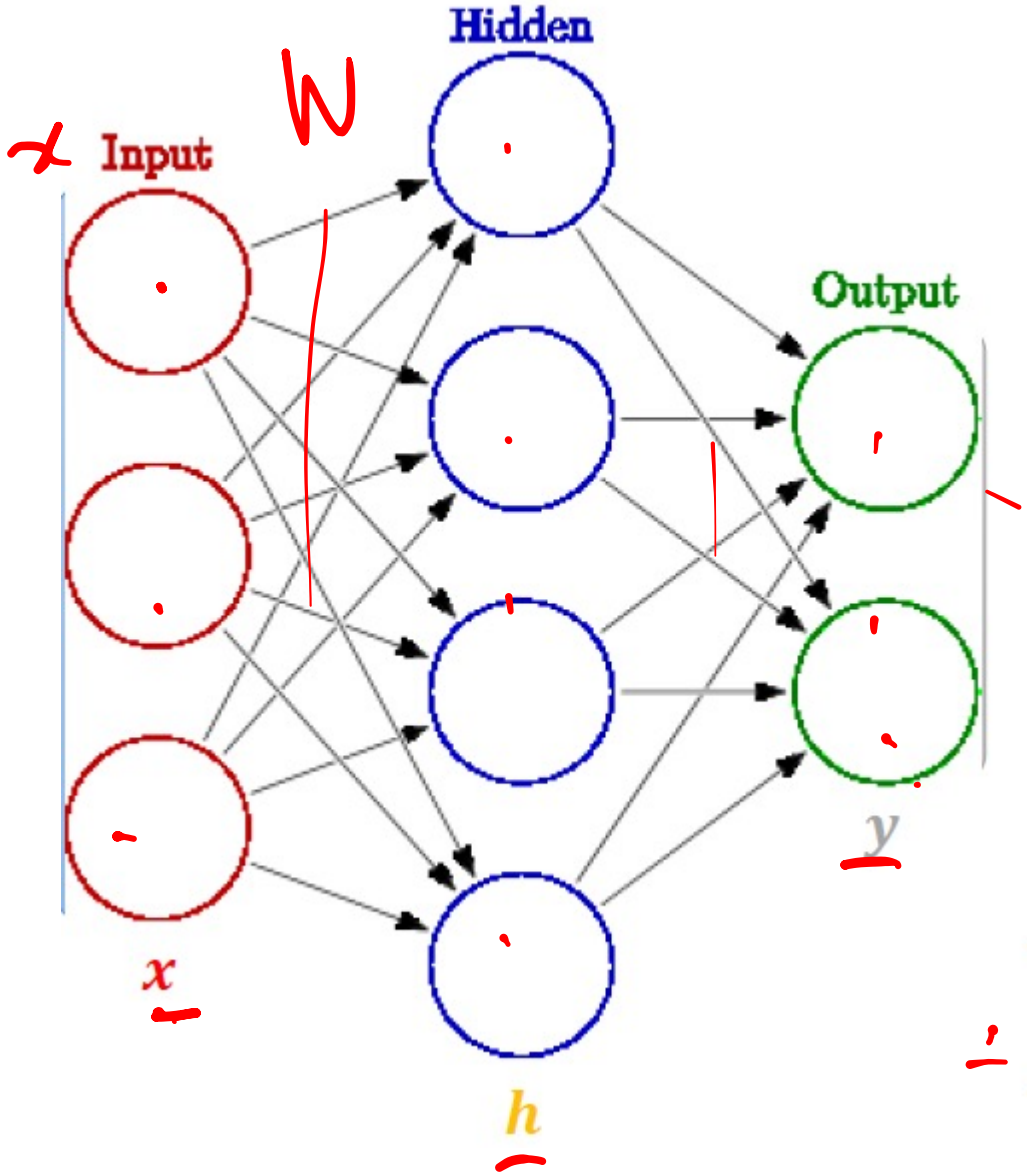
Stochastic Gradient Descent : It considers any one record for fitting a line & repeats this process for said no. of iterations




REPLAY

Trainer: Dr. Darshan Ingle.

Neural Network Learning Parameters



Weights

$h = \sigma(W_1 x + b_1)$

$y = \sigma(W_2 h + b_2)$

→ Activation function

$4 + 2 = 6$ neurons (not counting inputs)

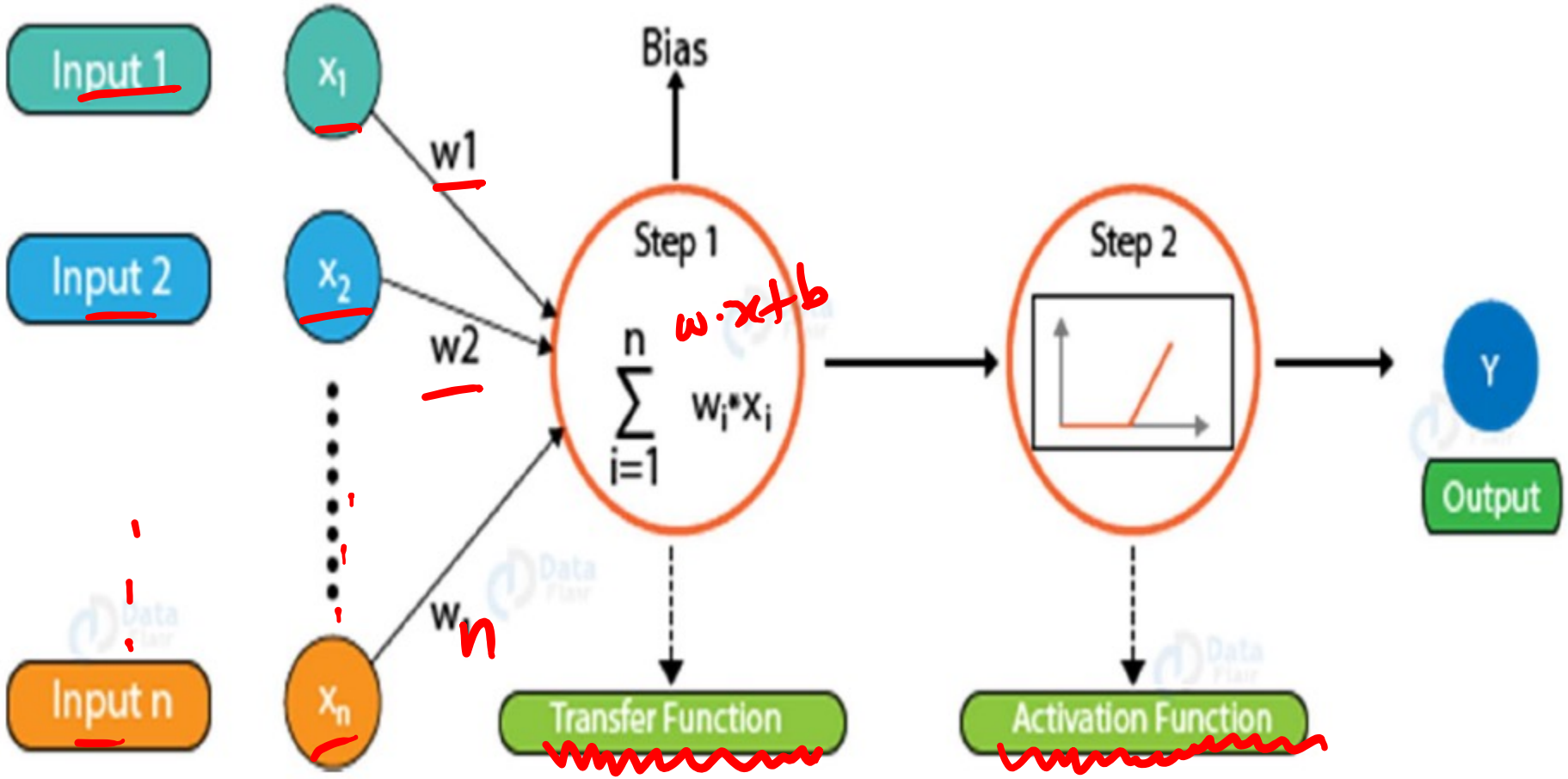
$[3 \times 4] + [4 \times 2] = 12 + 8 = 20$ weights

$4 + 2 = 6$ biases

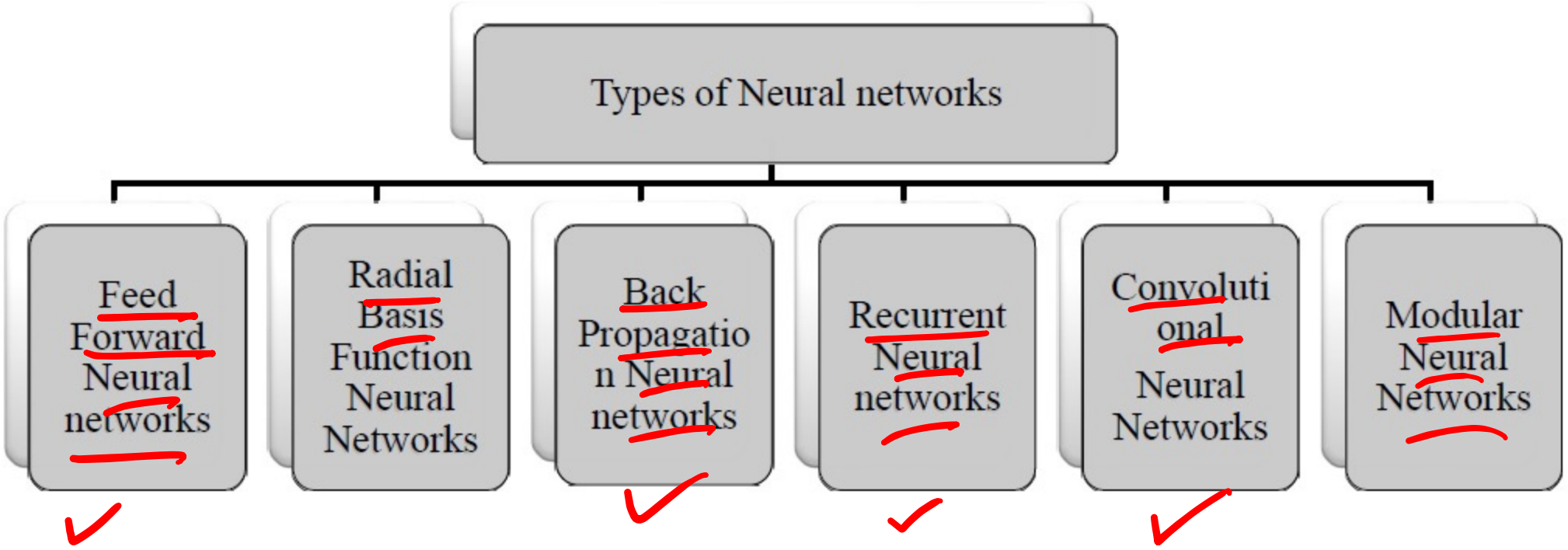
$\therefore 20 + 6 = 26$ Learnable parameters

More Terminologies of a NN

wx+b → Transfer Function



Types of Neural Network



In real life, weights are initialized RANDOMLY.

Forward and Backward Propagation

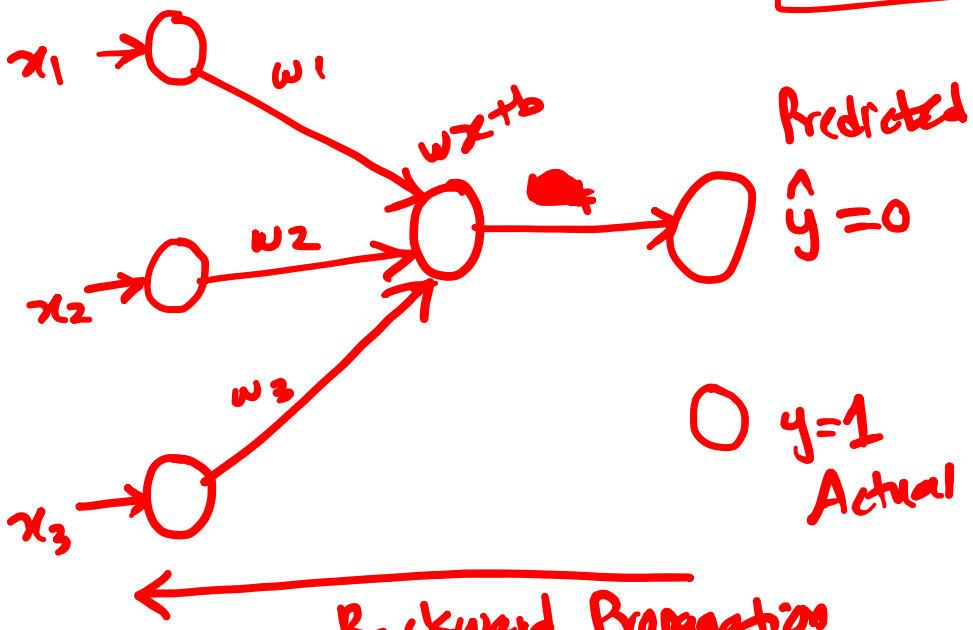
x_1 Res. Score	x_2 Proj. Score	x_3 Grade Score	o/p Grant Yes/No
80	70	83	1

$y = [w_1x_1 + w_2x_2 + w_3x_3] + b$

$z = \sigma(y)$ [Sigmoid]

Fwd Propagation →

1 EPOCH = 1FP + 1BP



Note: ① Single o/p Node = Loss Function

② Multiple o/p Node = Cost Function $\sum_{i=1}^{n} (y-\hat{y})^2$

Min. Loss ⇒ Optimizers

$Loss = (y-\hat{y})^2$

$= (1-0)^2$

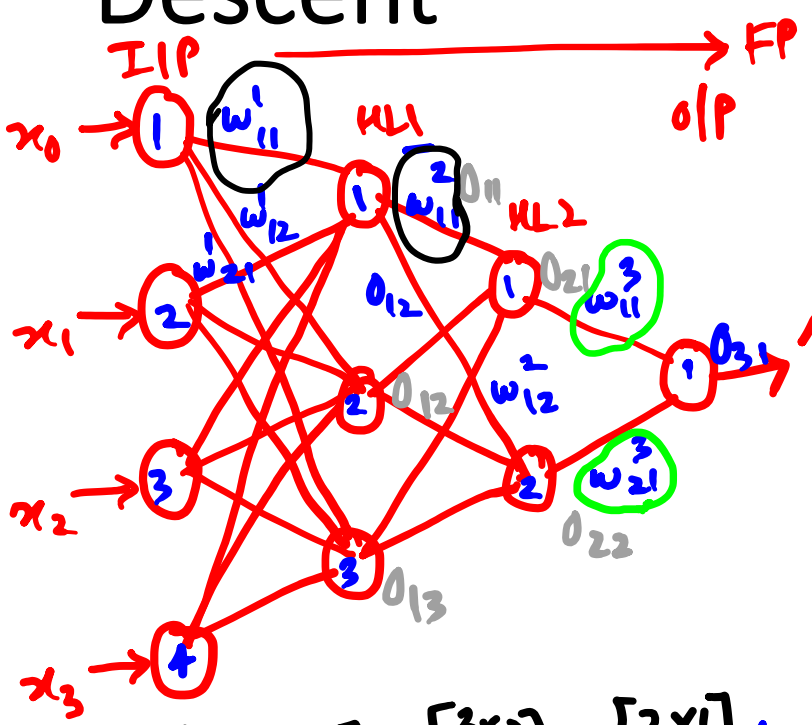
$= 1$

Formula!

$$w_{1_{new}} = w_{1_{old}} - \eta \cdot \frac{\partial L}{\partial w_{1_{old}}}$$

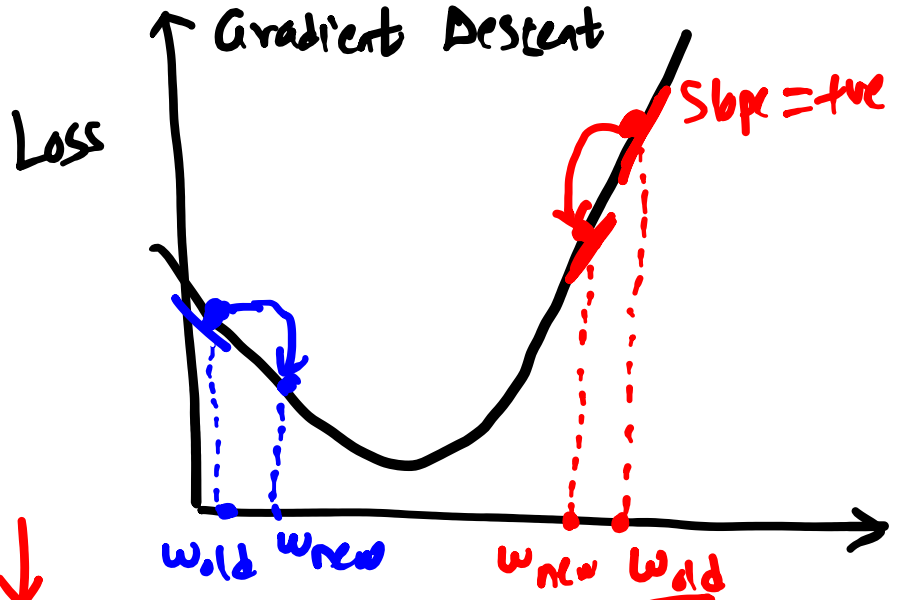
η = Learning rate. Assume $\eta = 0.01$ [0 - 1]

Multi Layer Neural Network Training w.r.t Gradient Descent



Predicted

$Loss = (y - \hat{y})^2 \downarrow$



$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w_{old}}$$

assume $\eta = 0.001$

Input Layer [1×3] [3×2] [2×1]

$$\begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w^1_{33} \\ w^1_{41} & w^1_{42} & w^1_{43} \end{bmatrix}_{4\times3} \quad \begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w^2_{31} & w^2_{32} \end{bmatrix}$$

$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w_{old}}$

$= w_{old} - \eta \cdot (-ve)$

$= w_{old} + (+ve)$

$w_{new} > w_{old}$

$= w_{old} - \eta \cdot (+ve)$

$= w_{old} - (+ve)$

$w_{new} < w_{old}$

CHAIN RULE IN BACKPROPAGATION

$$w^3_{11\,new} = w^3_{11\,old} - \eta \cdot \frac{\partial L}{\partial w^3_{11\,old}}$$

$$\frac{\partial L}{\partial w^3_{11\,old}} = \frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w^3_{11\,old}}$$

$\therefore$ we get $w^3_{11\,new}$ (updated weight)

Similarly, we can calculate for $w^3_{21\,new}$.

How to calculate $w^2_{11\,new}$?

$$w^2_{11\,new} = w^2_{11\,old} - \eta \cdot \frac{\partial L}{\partial w^2_{11\,old}}$$ Slope

$$\frac{\partial L}{\partial w^2_{11\,old}} = \left[\frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial w^2_{11\,old}}\right]$$

$$w^1_{11\,new} = w^1_{11\,old} - \eta \cdot \frac{\partial L}{\partial w^1_{11\,old}}$$

$$\frac{\partial L}{\partial w^1_{11\,old}} = \left[\frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial w^1_{11\,old}}\right] + \left[\frac{\partial L}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial O_{22}} \cdot \frac{\partial O_{22}}{\partial O_{12}} \cdot \frac{\partial O_{12}}{\partial w^1_{11\,old}}\right]$$