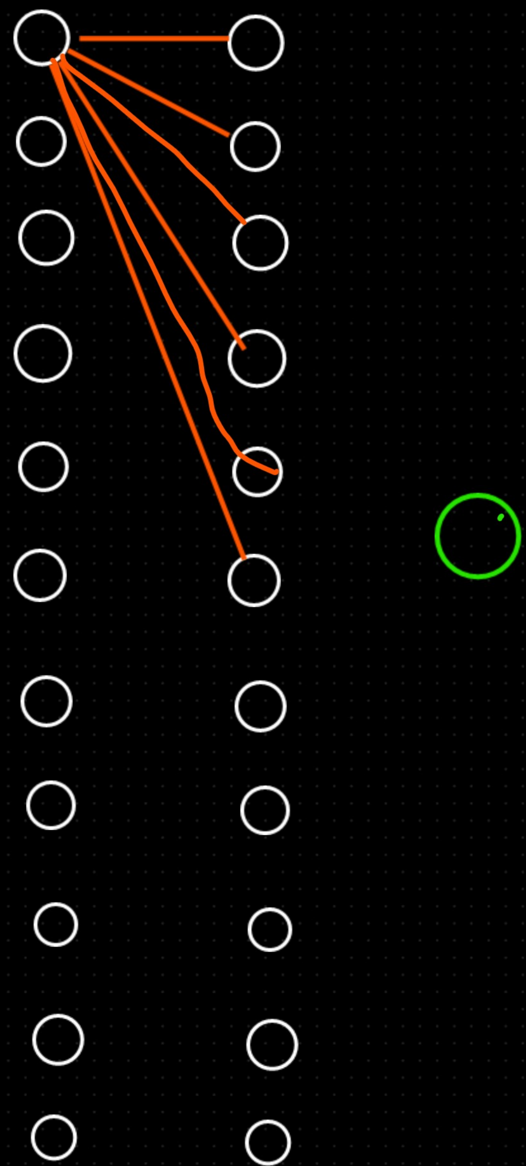
1. ANN Practical
2. Loss fn
3. BP
4. Vanishing gradient
5. Exploding gradient
6. Act fn / Loss / Optimizer
7. One more Practical
8. Normalization, Regularization
9. tensorflow → Regression
10. Pytorch - Regression

Start with ipynb that will give an overview about ANN implementation
AND AFTER THIS
At the end of this notes read about Loss Vs Cost fun^



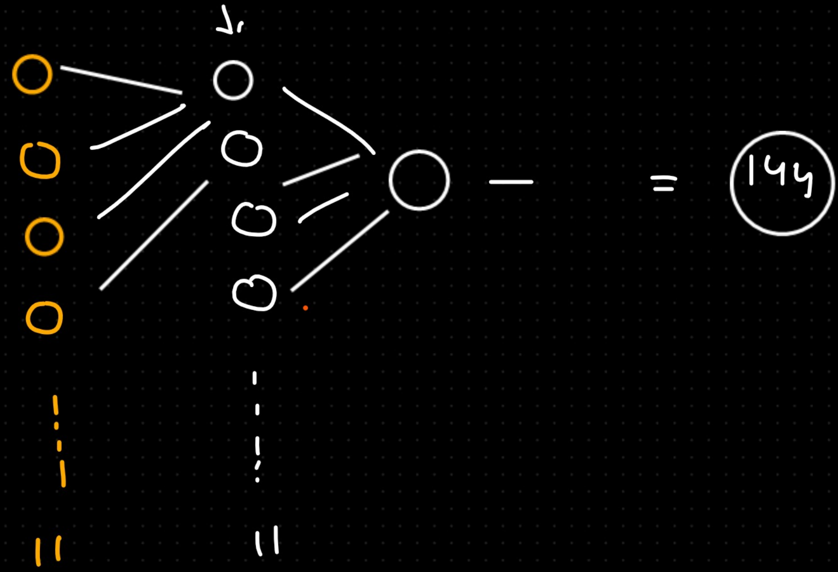
$11 \times 11 = 121 + 11 = 132$

$11 \times 1 + 1$

$11 + 1 = 12$

$$\begin{array}{r} 132 \\ 12 \\ \hline 144 \end{array}$$

```
model=Sequential()
model.add(Dense(11,activation="sigmoid",input_dim=11))#input layer
model.add(Dense(1,activation="sigmoid"))#output layer
```



$11 \times 11 = 121 + 11$

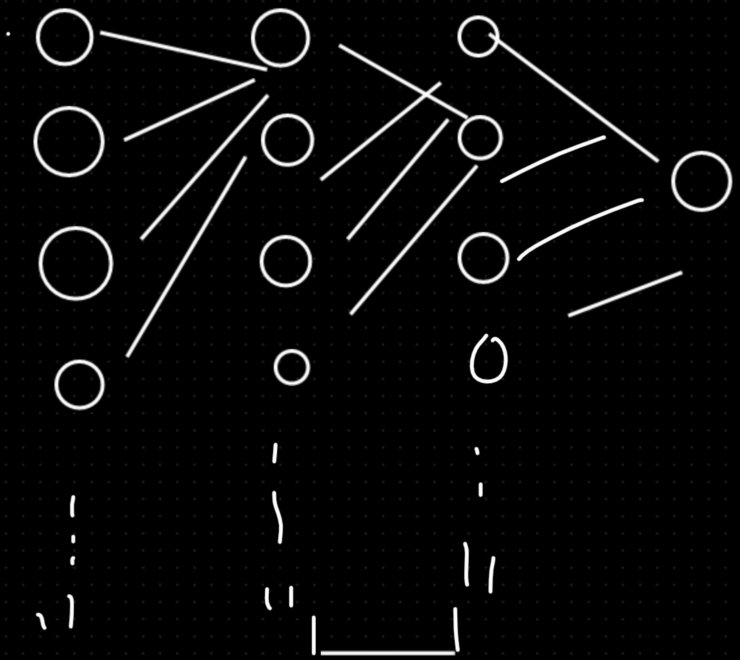
$= 132$

$11 \times 1 + 1$

$11 + 1 = 12$

$= 144$

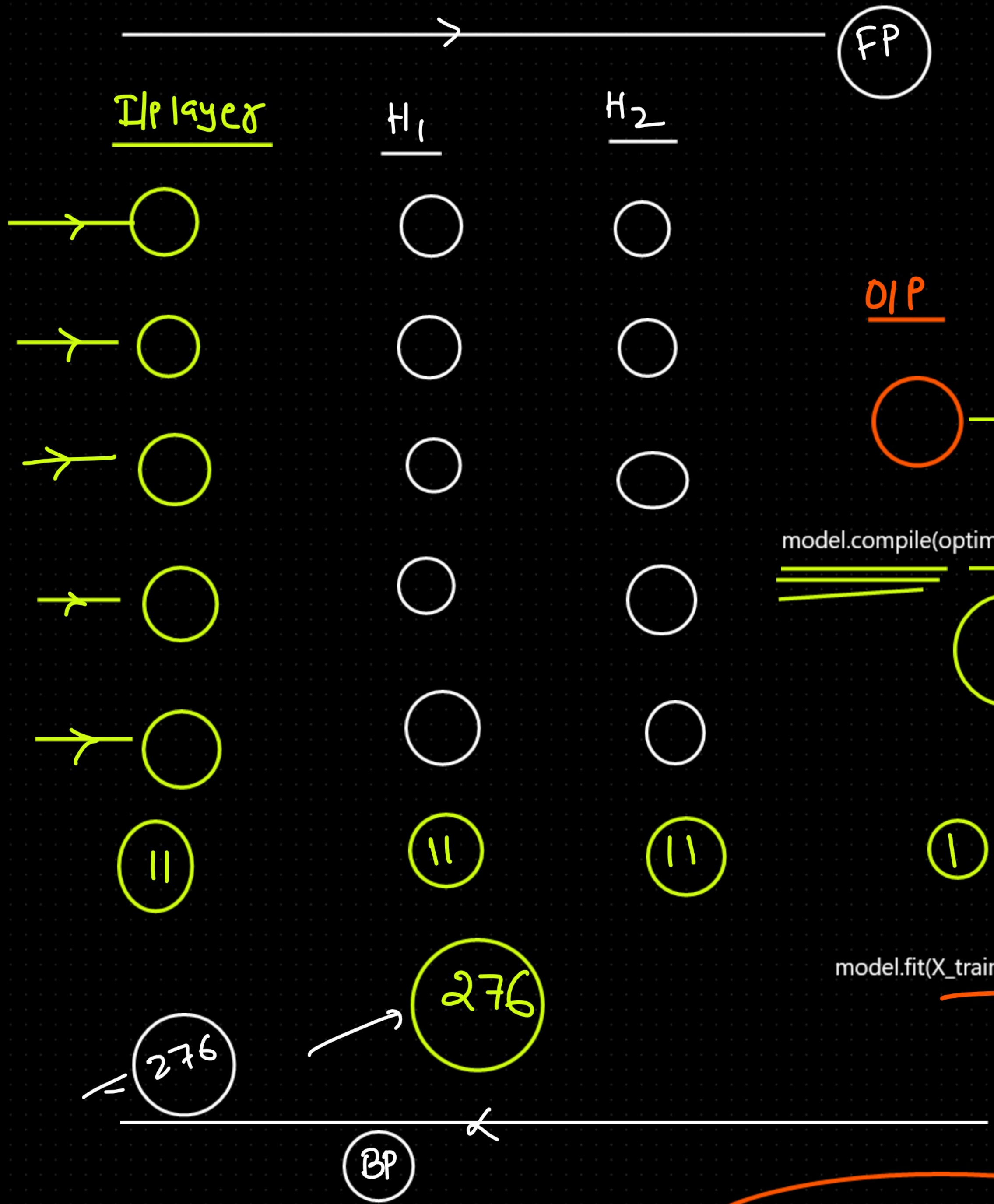
```
model=Sequential()
model.add(Dense(11,activation="sigmoid",input_dim=11))#input layer
model.add(Dense(11,activation="sigmoid"))#hidden layer
model.add(Dense(1,activation="sigmoid"))#output layer
```



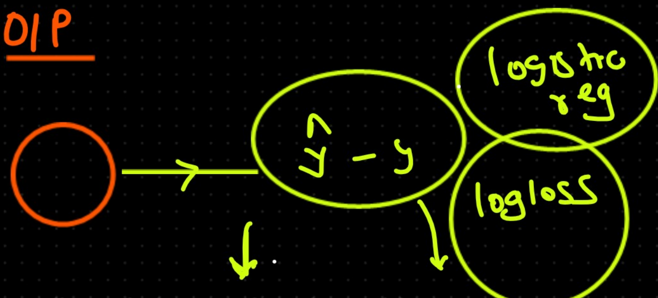
$$
\begin{array}{r}
132 \\
132 \\
12 \\
\hline
\rightarrow \underline{\underline{276}}
\end{array}
$$

$11 \times 11 + 11$

$121 + 11 = 132$ 132 12




```
model=Sequential()
model.add(Dense(11,activation="sigmoid",input_dim=11))#in
model.add(Dense(11,activation="sigmoid"))#hidden layer
```




```
model.compile(optimizer="Adam",loss="binary_crossentropy",metrics=["accuracy"])
```

BP

```
model.fit(X_train_scale,y_train,epochs=100,verbose=1)
```

1 epoch = 1 FP + 1 BP

FP

$\underline{(\hat{y} - y)}$

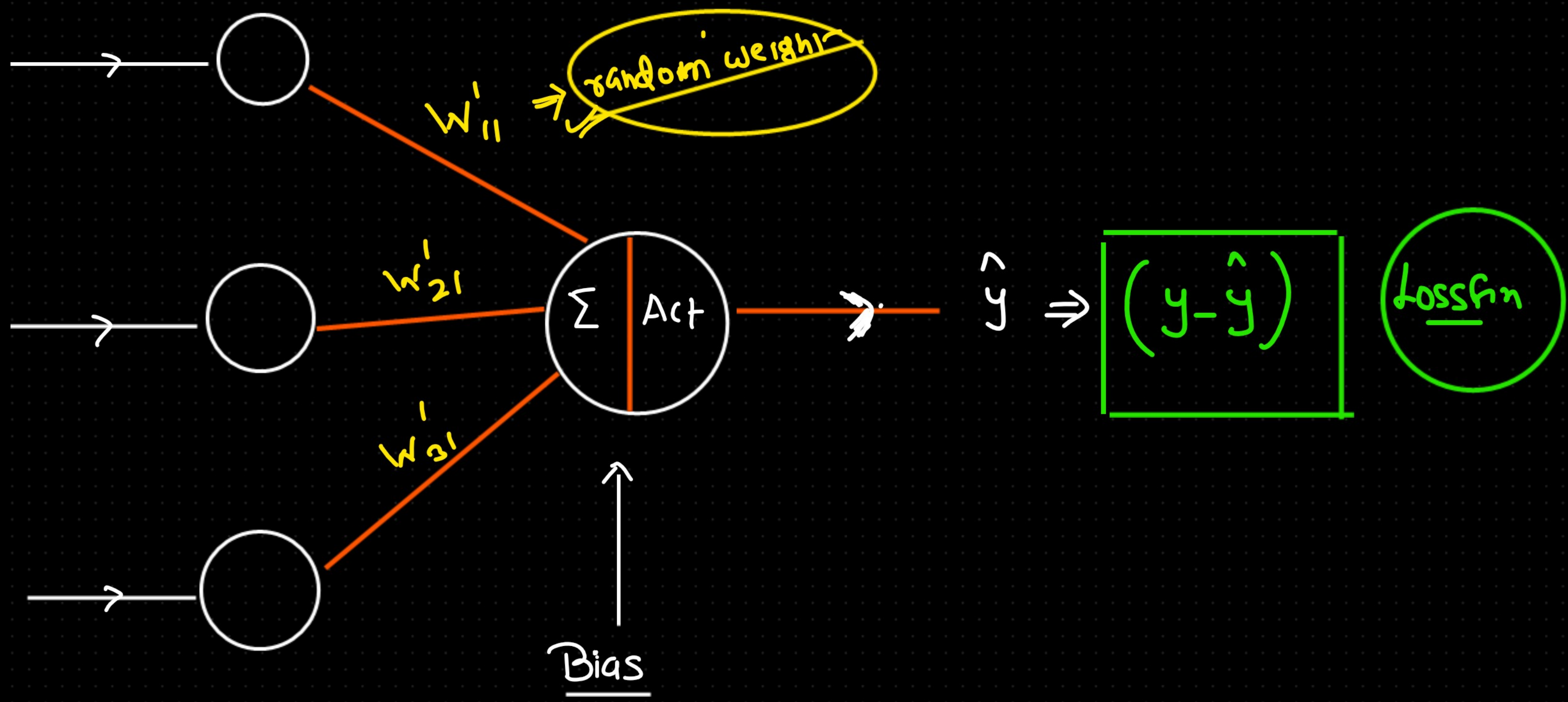
Loss

276

100times

- ✓ Loss function
- ✓ Backpropogation

forward Propogation = Calculating the value (output)



Backward Propogation

Update the Parameter ⇒ (weight, Bias)

Linear regression $\hat{y} = mx + c$

m = weight
C = Constant

⇒ I need to update it, how you will get to know?

Loss

Minimum loss

Without loss we cannot update weight and bias during Backward Propagation. If there is no loss then definitely there is no need to update weights as well as those respective weights will represent optimal weights give zero or negligible loss. This explains the importance of calculating loss during forward propagation

Loss v/s cost

- When actual - pred is calculated for only 1 record then loss fun^
- Whereas, when actual - pred is calculated for whole dataset then cost fun^

Height	Weight	BMI	BMI_Pred	
170	65	22	21	= 1
185	68	23	20.5	= 2.5
175	71	21	21.5	= 0.5

(actual - Pred) = Loss

$$\frac{1}{n}\sum_{i=1}^{n}(\text{actual} - \text{Pred}) \Rightarrow \text{Cost}$$

Loss = one observation
Cost = entire data

Types of loss fun^ in Regression based problem statement:

ANN

Regression

1. MSE ✓
2. MAE ✓

Types of loss fun^ in Classification based problem statement:

Classification

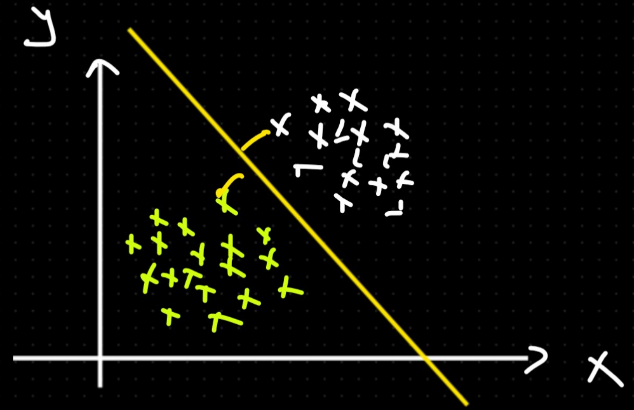
1. Binary cross entropy or log loss
2. { Categorical cross entropy
   Sparse categorical cross entropy }

MSE and MAE was already covered in ML when Krish sir was taking about different performance metrices to be used for regression based model. Go through that notes

MSE ⇒ Mean Squared error

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

Square

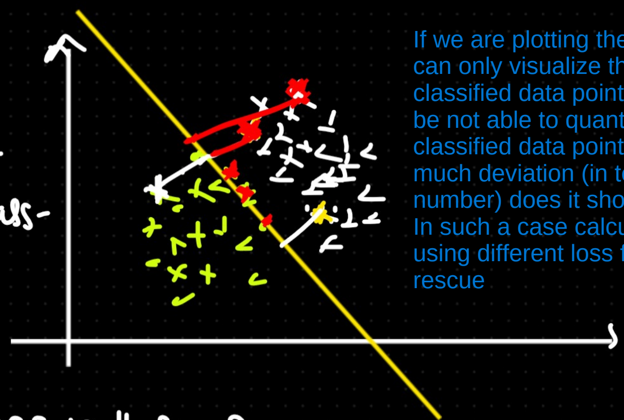


Eg :

BMI	Height	Weight (y=Target/Actual)	Pred_weight ($\hat{y}$=Predicted)		loss=y-y^
22	170	68 ⟷	67	=	1
20	160	70 ⟷	66	=	4
23	175	75 ⟷	80	=	-5
				=	0

Loss

How many points are wrong classified = 3



If we are plotting the graph then we can only visualize the miss classified data points. But we will be not able to quantify these miss classified data points that how much deviation (in terms of number) does it show from actual. In such a case calculating loss using different loss fun^ comes into rescue

How much loss is this : ? No.
But Quantity = ?

See here if we have not squared individual losses then net loss would be 0 which is false. So, in order to justify the loss we are squaring the individual losses so that net loss is not 0

68 66 = 2
70 72 = -2
75 76 = -1

= -1

Distance = Based on Distance
ho much loss

$\hat{y} = mx + c$

y

Squre ⇒ $1^2 + 4^2 + (-5)^2 = 1 + 16 + 25$
= 42

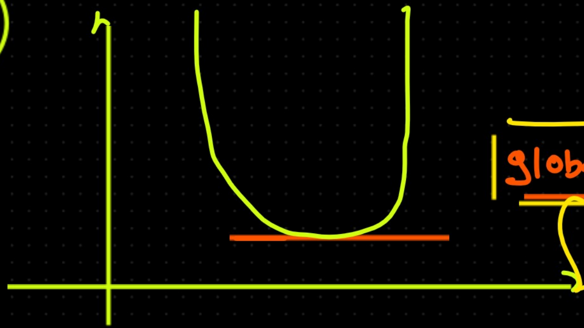
Mod (MAE)
-ve = +ve
|-ve| = +ve

1 ⇒ 1
4 ⇒ 16
25 = 625
⇒

10 ⟹ 100

MSE ⇒ quardatic eq

In MSE since we are squaring so that's why convex fun^(parabolic) is formed which is differentiable hence gradient descent can be applied.



Advantage ⇒ 1 Convex fn ⇒
2 easy to interpret

global minima
derivative also at every point

gredient decent ⇒ differentiation

$$M_{new} = M_{old} - \eta \frac{\partial L}{\partial m} \quad ?$$

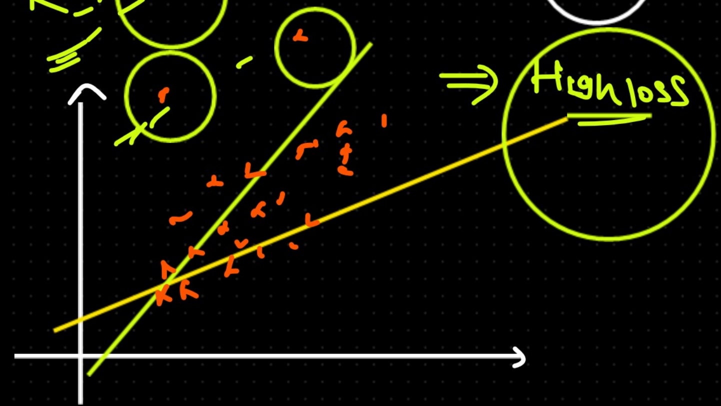
loss → MSE

not robust with

Disadvange ⇒ (1) outlier

(2) real value ⇒ inverse the value

$4 \Rightarrow 4^2 = 16 \Rightarrow$ Square $\sqrt{16} \Rightarrow 4 \leftarrow$ Actual loss



$= y - \hat{y} \Rightarrow$ updation

MAE ⇒ $$\boxed{\frac{1}{n}\sum_{i=1}^{n} |y-\hat{y}|}$$

⇓

Mode

Height	weight	BMI	Pred_BMI	
170	65	22 ↔	21	= 1
180	60	21 ↔	24	= -3
185	61	19 ↔	20	= -1

$|1| = 1$

$|-3| = 3$

$|-1| = 1$

$\boxed{1+3+1 = 5}$

MSE

Loss ⇒ weight

Advantage ⇒ ① giving me a value in range

② robust to the outliers

Disadvange ⇒ $\boxed{\text{We can not GD}}$ (use)

since convex function is not formed when we use mod

⇐ it is not convex fn

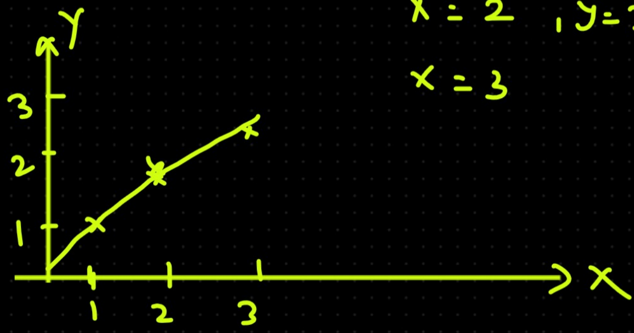
Mode fn ⇒ it is not a convex fn

← this is not Differensia-ble

$\boxed{y \Rightarrow f(x) \Rightarrow x}$ $\Rightarrow$ $\boxed{Y = X}$

↑

functions
limits
derivation
Integration



$x = 1\ ,\ y = 1$

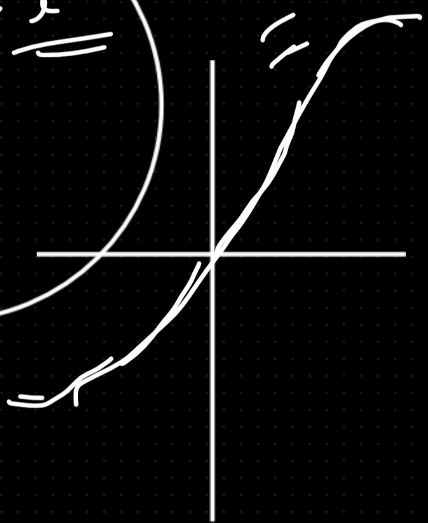
$x = 2\ ,\ y = 2$

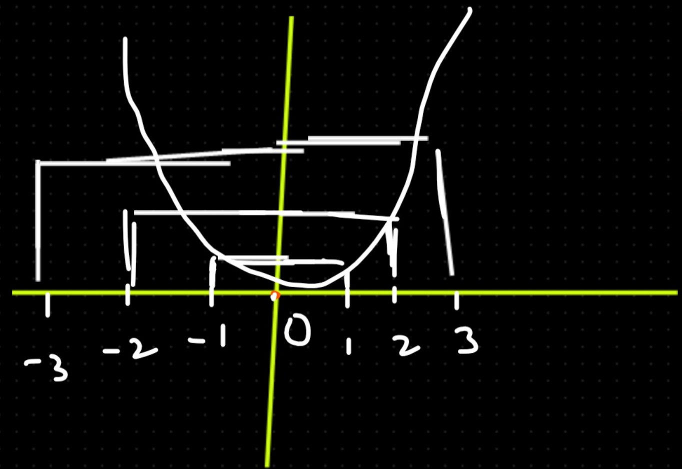
$x = 3$

$\boxed{y = f(x) = x}$

$= \boxed{y = f(x) = x^2}$

$y = f(x) = x^3$





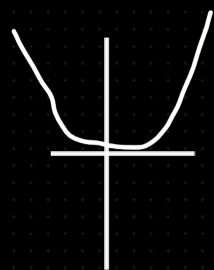
$y = 0$

$y = f(2) = 4$

$y = f(3) = 9$

$f(-2) = 4$

$f(-3) = 9$



$x^2 \neq (y - \hat{y})^2$

Quadratic

Parabola

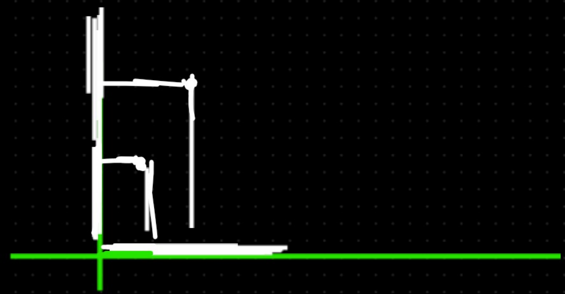
$\frac{dy}{dx} = \frac{Y}{X} = x^2 = dx$

$$m_{new} = m_{old} - \eta \frac{\partial L}{\partial w}$$

$$\frac{dy}{dx} = x=1$$

$x=1 \quad = 2\times 1 = 2$

$x=2 \quad = 2\times 2 = 4$



$$\frac{dx}{dy} = 1$$