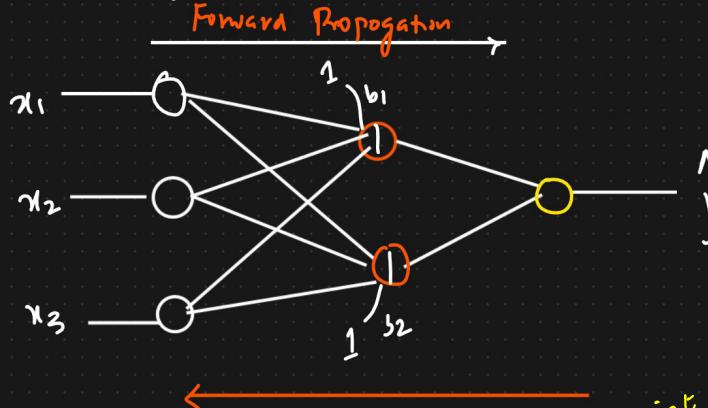


Loss function And Cost Function



loss function → single datapoint
Error

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

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

\uparrow Error
 \downarrow

$x_1 \quad x_2 \quad x_3 \quad \text{O/P}$

\hat{y}

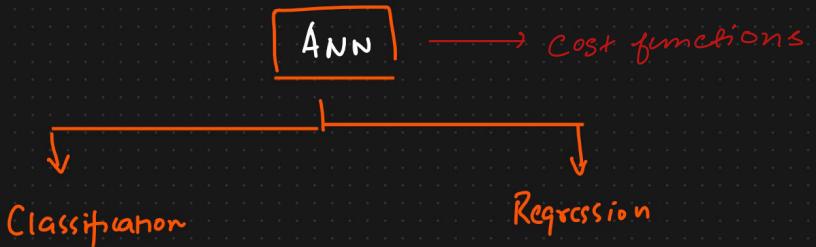
$\boxed{\text{loss} = (y - \hat{y})^2}$

$\left\{ \begin{array}{l} \text{--- --- 0} \\ \text{--- --- 1} \\ \text{--- --- 0} \\ \text{--- --- 1} \end{array} \right.$

Gradient Descent

Cost function → entire Data
error.

Cost function → loss function
for Batch's
of Data
Points

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


- ① Mean Squared Error (MSE)
- ② Mean Absolute Error (MAE)
- ③ Huber loss
- ④ RMSE

① Mean Squared Error (MSE)

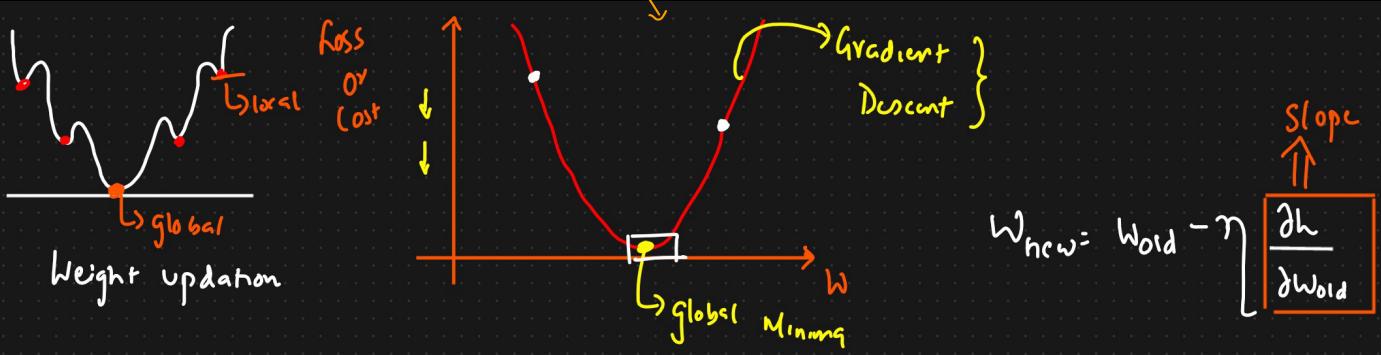
$$\text{loss function} = (y - \hat{y})^2$$

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

↳ Quadratic Equations

If we plot Loss function
we get Gradient Descent





Advantages

① MSE is Differentiable

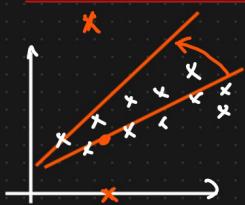
Imp: ② It has 1 local OR global Minima

③ It converges faster

Disadvantages

Imp:

① Not Robust to outliers



penalizing
the error

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

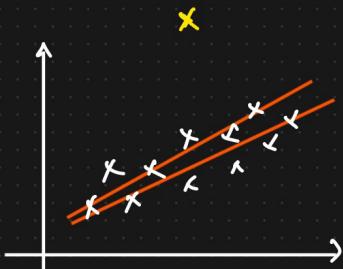
② Mean Absolute Error (MAE)

$$\text{Loss fn} = |y - \hat{y}|$$

$$\text{Cost fn} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

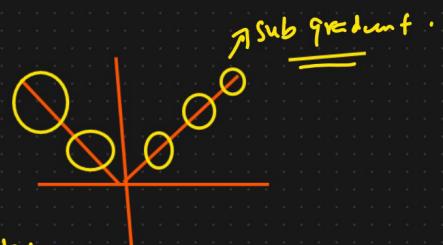
Advantages

① Robust to outliers



Disadvantages

① Convergence usually takes
time in MAE



③ Huber loss

Combination of MSE and MAE where we use MSE for no outliers and vice versa.

① MSE

② MAE

MSE

No outlier

Hypoparameter

$$\text{Cost fn} = \begin{cases} \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \leq \boxed{s} \\ \boxed{s} & \text{if } |y_i - \hat{y}_i| > \boxed{s} \end{cases}$$

$$\begin{cases} \delta |\hat{y} - y| - \frac{1}{2} \delta^2, & \text{otherwise} \\ \downarrow \\ \text{MAE} \end{cases}$$

④ RMSE (Root Mean Squared Error)

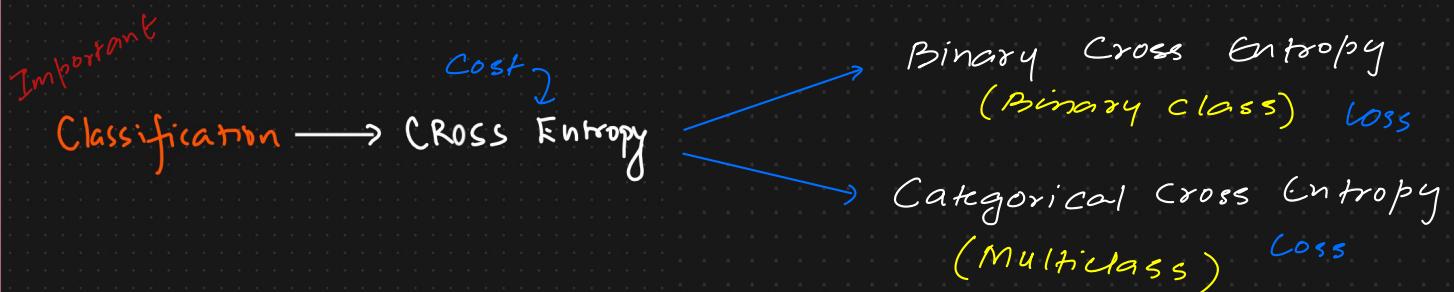
=

$$\text{Cost function} = \sqrt{\frac{\sum_{i=1}^N (\hat{y}_i - y_i)^2}{N}}$$

Advantages

Disadvantages

② Loss or Cost function For Classification Problems



① Binary Cross Entropy

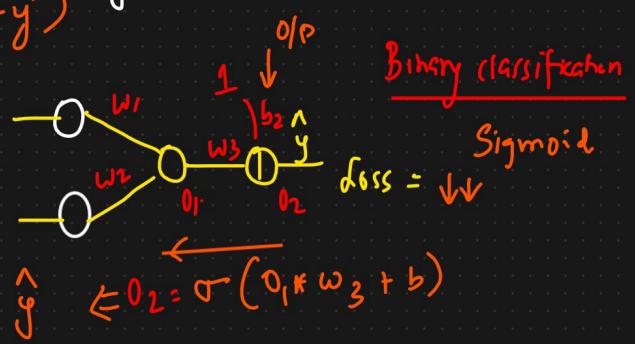
log loss

$$\text{loss} = -y * \log(\hat{y}) - (1-y) * \log(1-\hat{y})$$

$$\text{loss} = \begin{cases} -\log(1-\hat{y}) & \text{if } y=0 \\ -\log(\hat{y}) & \text{if } y=1 \end{cases}$$

y = Actual Value

\hat{y} = Predicted Value



$$\hat{y} = \frac{1}{1+e^{-z}} \Rightarrow \text{Sigmoid Activation function}$$

② Categorical Cross Entropy (Multiclass classification) ONE is used

$\xrightarrow{\text{ONE} \rightarrow \text{One Hot Encoding}}$

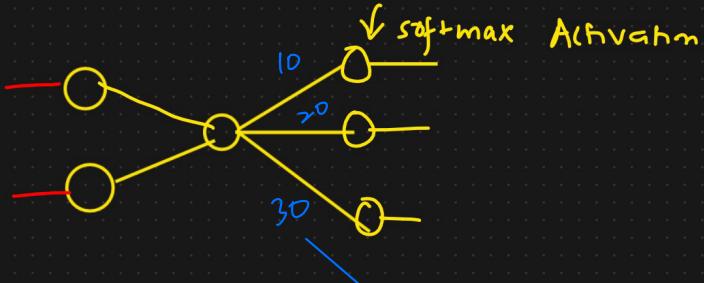
f_1	f_2	f_3	O/P	$j=1$ Good	$j=2$ Bad	$j=3$ Neutral	$C = \text{No. of categories}$
$\rightarrow 2$	3	4	Good	1	0	0	i = 1 to n
$\rightarrow 5$	6	7	Bad	0	1	0	
$\rightarrow 8$	9	10	Neutral	0	0	1	

$$d(x_i, y_i) = - \sum_{j=1}^C y_{ij} * \ln(\hat{y}_{ij})$$

Actual value $\leftarrow y_{ij} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \dots & y_{1c} \\ y_{21} & y_{22} & y_{23} & \dots & y_{2c} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & y_{n3} & \dots & y_{nc} \end{bmatrix}$

$$y_{ij} = \begin{cases} 1 & \text{if the element is in the class} \\ 0 & \text{Otherwise} \end{cases}$$

Prediction $\leftarrow \hat{y}_{ij} \Rightarrow \text{Softmax Activation} = \text{Soft}(z) =$



$$z_i \\ e^{z_i} \\ \sum_{j=1}^k e^{z_j}$$

$$\begin{aligned}
 & \text{O/p } y_j = \text{Probabilities} \\
 & \text{Categorical} \Rightarrow [0.2, 0.3, 0.5] \\
 & \text{Cross Entropy} \\
 & \quad \downarrow \\
 & \quad \boxed{\text{This also gives the probability of other categories}}
 \end{aligned}$$

$$\begin{aligned}
 & [0.1, 0.2, 0.3, 0.2, 0.2] \leftarrow 1 \\
 & \quad \downarrow \\
 & = \frac{e^{z_i}}{e^{z_j}} \\
 & = \frac{e^{10}}{e^{10+20+30}}
 \end{aligned}$$

③ Sparse Categorical Cross Entropy

$$\begin{array}{c}
 \xrightarrow{\text{0 } 1^{\text{st}} \text{ } 2^{\text{nd}}} \text{Categories} \\
 \boxed{[0.2, 0.3, 0.5]} \\
 \downarrow
 \end{array}$$

Disadvantage

- ① losing info about the probability of other category.

$$\boxed{2^{\text{nd}} \text{ Index}} \Rightarrow \underline{\text{O/p}}$$

$$\begin{array}{c}
 \boxed{0 \ 1 \ 2^{\text{nd}} \ 3^{\text{rd}} \ 4^{\text{th}}} \\
 \underbrace{[0.2, 0.3, 0.1, 0.2, 0.2]}_{\text{Index}} \Rightarrow 1^{\text{st}} \text{ Index} \\
 \downarrow \\
 \text{Category} \Rightarrow \underline{\text{O/p}}
 \end{array}$$

④ Right Combination

	Activation applied	Hidden Layers	O/p Layer	Problem Statement	Loss function
①	Sigmoid	ReLU or its Variants		Binary Classification	Binary Cross Entropy
②	Softmax	ReLU or its Variants		Multi Class	Categorical or Sparse CE
③	Linear	ReLU or its Variants		Regression	MSE, MAE, Huber loss, RMSE

Conclusion

✓ Types of Neural Networks & Their Use Cases

Neural Network Type	Best Suited For	Data Type
ANN (Artificial Neural Network)	Regression & classification on tabular data	Numeric/tabular data
CNN (Convolutional Neural Network)	Image recognition, video analysis, object detection	Images, video, spatial data
RNN (Recurrent Neural Network)	Sequence modeling, text, time-series forecasting	Text, speech, time-series

- Loss function → error for single data point
- Cost function → error for entire Data (epoch or batch)

Regression



Loss → MSE → outlier sensitive
 MAE → outlier insensitive
 Huber loss → comb MSE/MAE
 RMSE

Q: Why do we need Regression Activator func in LR?
 Unlike softmax, it is used to output raw unbounded values, not non-linearity.

Q: "No Activation" = Linear Activation

That means:

The output layer just returns the raw value from the final neuron, with no transformation applied.

✓ Summary:

Term	Meaning
No Activation	Return raw output (linear function)
Used in	Regression (real-valued outputs)
Why	Keeps values unbounded ($-\infty$ to $+\infty$)

Classification

Classification Cheat Sheet

Classification Type	Loss Function	Activation Function
Binary Classification	Binary Cross Entropy (Log Loss) $L = -[y \cdot \log(p) + (1-y) \cdot \log(1-p)]$	Sigmoid $\sigma(x) = 1 / (1 + e^{-x})$
Multi-Class Classification	Categorical Cross Entropy $L = -\sum y_i \cdot \log(p_i)$ (for i = 1 to C)	Softmax $p_j = e^{z_j} / \sum e^{z_i}$ (for j = 1 to C)

Important

✓ Activation Functions by Problem Type

Problem Type	Hidden Layer Activation	Output Layer Activation
Binary Classification	ReLU	Sigmoid
Multi-Class Classification	ReLU	Softmax
Regression (Single Output)	ReLU	Linear

\$1M Question ?

"Why should I use **Artificial Neural Networks (ANNs)** for regression or classification when I can just use **Linear Regression** or **Logistic Regression**?"

Short Answer:

You don't always need ANN. But you might want to use ANN when **traditional ML models aren't enough**.

Deeper Comparison:

Aspect	Linear/Logistic Regression	ANN (Deep Learning)
Complexity of patterns	Handles linear relationships only	Captures non-linear relationships
Feature engineering	Often requires manual feature crafting	Learns features automatically
Scalability	Fast and interpretable	Scales better with large data
Overfitting risk	Low on small data	High unless regularized/trained well
Interpretability	Very high (coefficients are clear)	Low (black box)
Speed	Very fast	Slower, needs more compute

Example Scenario:

Logistic Regression Fails:

You have a classification problem where the classes are **not linearly separable** (e.g., spiral or concentric circles).

👉 Logistic regression draws a straight line. It fails.

ANN Wins:

A neural network can learn **non-linear decision boundaries** and separate the classes better.

TL;DR:

Use **Logistic/Linear Regression** when:

- "Data is small"
- "Relationships are linear"
- "You want speed & interpretability"

Use **ANN** when:

- "Data is large or complex"
- "You suspect non-linear interactions"
- "You're hitting performance limits with traditional models"



Pro tip: Always **start simple** (like Logistic Regression).

Only add complexity (ANN) when it's needed.