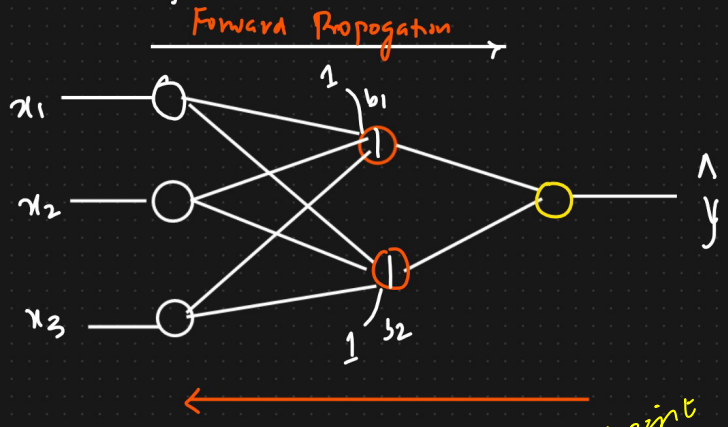
Loss function And Cost Function



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

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

Optimizers

Gradient Descent

x_1	x_2	x_3	O/P ($\hat{y}$)
-	-	-	0
-	-	-	1
-	-	-	0
-	-	-	1

Loss function → Single Datapoint Error

Costfunction → Entire Data Error. → Loss function for Batches of Data Points

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

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

ANN → Cost functions

- Classification
- Regression
  1. Mean Squared Error (MSE)
  2. Mean Absolute Error (MAE)
  3. Huber Loss
  4. RMSE

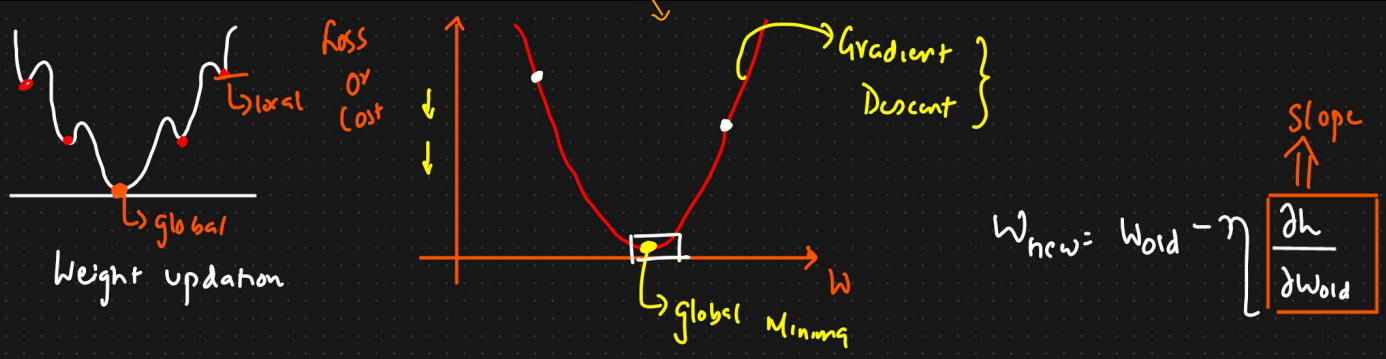
1. Mean Squared Error (MSE)

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

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

↳ Quadratic Equations

If we plot loss function we get Gradient Descent



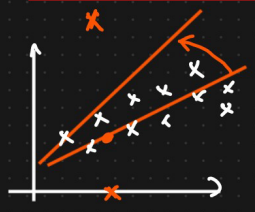
Advantages

1. MSE is Differentiable
2. It has 1 local or global Minima
3. It converges faster

Disadvantages

Imp:

1. Not Robust to outliers



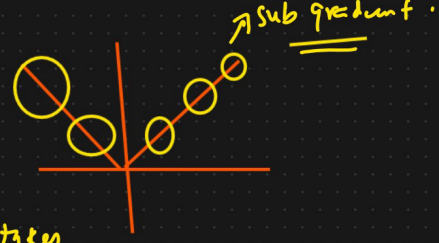
penalizing the Error

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

2. Mean Absolute Error (MAE)

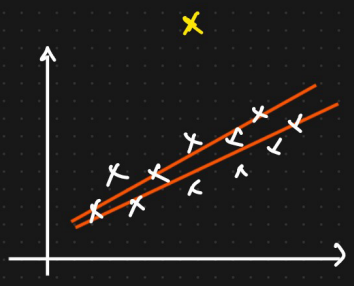
Loss fn : $|y - \hat{y}|$

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

Imp → MSE → prone to outlier
MAE → Not prone

3. Huber loss

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

1. MSE
2. MAE

Cost fn = $\begin{cases} \frac{1}{n}\sum_{i=1}^{n}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \le \boxed{\delta} \end{cases}$

MSE ⇑ NO outlier ⇑ Hyperparameter

$$\left|\delta|y-\hat{y}| - \frac{1}{2}\delta^2\right|, \text{otherwise}$$

⇓ MAE

↳ Based on hyper parameter value.

④ RMSE (Root Mean Squared Error)

$$=$$

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

Advantages

Disadvantages

② Loss or Cost function For Classification Problems

Important

Classification ⟶ Cross Entropy ← Cost

- Binary Cross Entropy (Binary class) Loss
- Categorical Cross Entropy (Multiclass) Loss

① Binary Cross Entropy

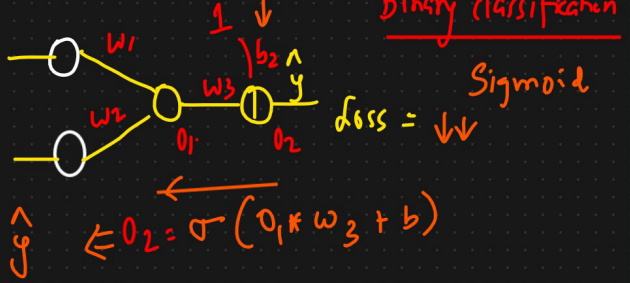
Log Loss ⇓

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

y = Actual Value

$\hat{y}$ = Predicted Value

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



Binary classification

Sigmoid

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

② Categorical CROSS Entropy (Multiclass classification) OHE is used

OHE → One Hot Encoding

	f_1	f_2	f_3	O/P	Good (j=1)	Bad (j=2)	Neutral (j=3)
→	2	3	4	Good	1	0	0
→	5	6	7	Bad	0	1	0
→	8	9	10	Neutral	0	0	1

C = No of categories

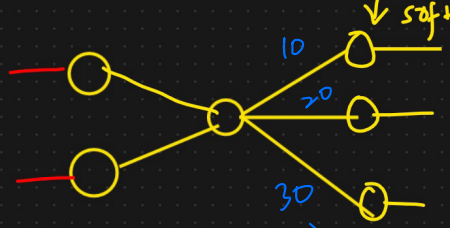
i = 1 to n

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

Actual ⇐ $y_{ij} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \cdots & y_{1C} \\ y_{21} & y_{22} & y_{23} & \cdots & y_{2C} \\ \vdots & & & & \end{bmatrix}$

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

Prediction ⇐ $\hat{y}_{ij}$ ⇒ Softmax Activation $= Sof(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$



O/p $\hat{y}_{ij}$ = Probabilities

Categorical Cross Entropy $\Rightarrow [0.2, 0.3, 0.5]$

[This also gives the probability of other categories]

$[0.1, 0.2, 0.3, 0.2, 0.2] \leftarrow 1$

$= \frac{e^{z_i}}{e^{z_j}}$

$= \frac{e^{10}}{e^{10+20+30}}$

③ Sparse Categorical Cross Entropy

0 1st 2nd → Categories

$[0.2, 0.3, 0.5]$

$\Downarrow$

2nd Index $\Rightarrow$ O/p

Disadvantage

① Loosing info about the probability of other category.

0 1 2nd 3rd 4th

$[0.2, 0.3, 0.1, 0.2, 0.2] \Rightarrow 1^{th}$ Index

$\Downarrow$

Category $\Rightarrow$ O/p

④ Right Combination

Activation applied

$\Downarrow$

	Hidden Layers	O/p Layer	Problem Statement	Loss function
①	Relu or its Variants	Sigmoid	Binary classification	Binary Cross Entropy
②	Relu or its Variants	Softmax	Multi class	Categorical or Sparse CE
③	Relu or its Variants	linear	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

NLP

Loss function → Error for single data point

Cost function → Error for entire data (epoch or Batch)

ANN → Regression
Classification

Regression

Loss → MSE → Outlier Sensitive
MAE → Outlier Insensitive
Huber Loss → Comb MSE / MAE
RMSE

VVimp Why do we need Regression Activator fn. in LR?
Unlike softmax, it is used to output Raw unbounded values, not non linearity.

🚫 "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 (−∞ to +∞)

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** 📌 $p_i = e^{(z_i)} / \sum e^{(z_j)}$ (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

$IM 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.