Loss function used in Neural Networks

Loss functions are used to measure the differences between the model's predicted output and the actual target value. These functions guide the optimization process, allowing the model to improve its performance. There are several loss functions commonly used in different types of deep learning tasks, each with specific use cases, advantages and limitations. Some of these are—

① Mean Squarred Error (MSE) ⇒

These are used in regression use cases where the prediction needs to be made on continuous data such as prediction of house prices, etc. It penalizes the larger errors more heavily because of the squared term and thus it is sensitive to outliers. The MSE function is differentiable at all points.

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

② Mean Absolute Error (MAE) ⇒

These also used in regression use cases. These are less sensitive to outliers because it doesn't square the errors. It is used in cases where robustness to outliers is needed. Unlike MSE, large errors are not penalized as heavily.

$$\boxed{MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|}$$

It is not differentiable at 0, because of which it is not easily optimisable.

③ Huber Loss ⇒

It is also used in regression usecases. It is a hybrid of both MSE and MAE which helps to reduce the effect of outliers. It balances the sensitivity towards outliers by combining MSE for small errors and MAE for large errors, while still mitigating the ~~st~~ extreme impact of outliers penalizing errors. The conditional nature of Huber loss formula makes it slightly harder to optimize.

$$\boxed{L_\delta(x) = \begin{cases} \frac{1}{2}x^2 & \text{for } |x| \le \delta \\ \delta\left(|x| - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases}}$$

here, $x = |y - \hat{y}|$, δ is a hyperparameter.

④ Cross Entropy Loss (Log Loss) ⇒

This loss function is used in classification use cases. There are two types of cross entropy loss— (a) binary ~~categorical~~ cross entropy
(b) categorical cross entropy.

Binary cross entropy is used for binary classification problems along with sigmoid activation function whereas categorical cross entropy is used with multi-class classification problem along with softmax activation function.

Cross entropy measures the divergence between the predicted probability distribution and the true distribution, making it effective for classification tasks.

Binary Cross Entropy Loss (binary classification)	$= -y \log(\hat{y}) - (1-y)\log(1-\hat{y})$
Categorical Cross Entropy Loss (multi-class classification)	$= -\sum_{c=1}^{c=K} Y_c \log(\hat{y}_c)$

** Categorical cross entropy expects the labels to be one-hot encoded.**

⑤ **Sparse Categorical cross entropy** ⇒ It is same as categorical cross entropy except it accepts the labels in numerical integer format and not one-hot encoded format.

⑥ **Hinge Loss** ⇒ This is used in binary classification with SVMs and it expects the labels to be either −1 or 1. If 0 or 1 is provided, it converts it back to −1 & 1. It is helpful in maximizing the margin between classes, improving generalization. Unlike cross-entropy loss, hinge-loss doesn't work when we need probabilistic outputs.

$$\boxed{L = \sum_{i=1}^{n} \max(1 - y \cdot \hat{y}, 0)}$$

⑦ **Focal Loss** ⇒ It is useful in imbalanced classification problems. Focal loss adjust the loss for hard-to-classify examples, thus preventing easy examples from overwhelming the loss calculation.

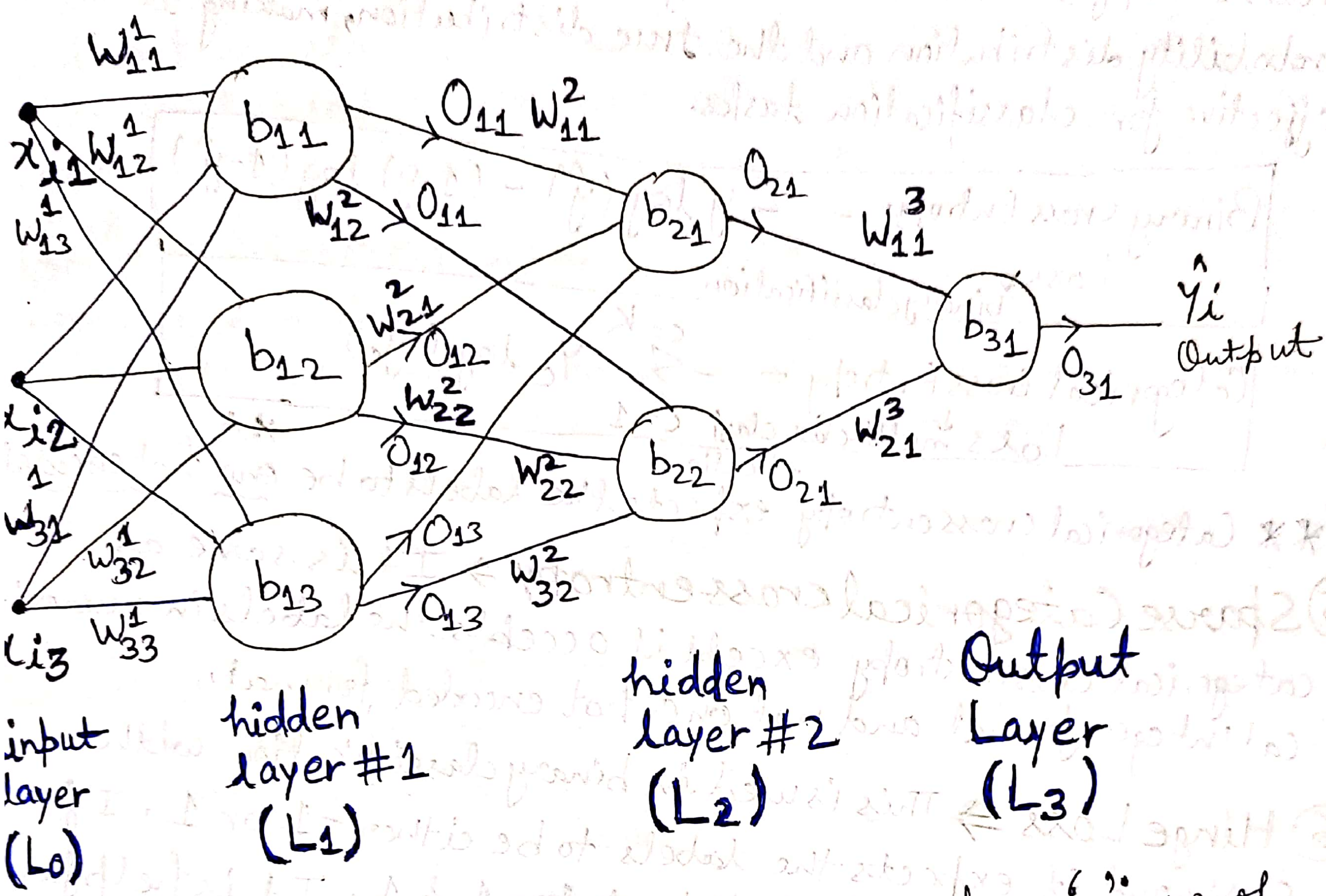
$$\boxed{L = -\alpha (1-\hat{y}_i)^{\gamma} y_i \log(\hat{y}_i)}$$

α & γ are hyperparameters.

These hyperparameters require tuning which makes training more complex.

Multi Layer Perceptron

Notation in MLP :-



Suppose, our dataset has $(m \times n)$ shape, where 'm' is no. of rows & 'n' is no. of columns. In the diagram, shown above we have 3 columns, i.e., $n=3$, therefore there are 3 inputs. 'i' denotes the 'i^{th}' row in the data.

Now, we can find out the number of trainable parameters in our MLP or neural network as shown below.

Between Layer L_0 & L_1, there are $3 \times 3 = 9$ weights and 3 biases for each neuron. $\Rightarrow 9+3 = 12$

Between Layer L_1 & L_2, there are $3 \times 2 = 6$ weights and 2 biases for each neuron of layer L_2. $\Rightarrow 6+2 = 8$

Between layer L_2 & L_3, there are $2 \times 1 = 2$ weights and 1 bias for neuron in L_3. $\Rightarrow 2+1=3$

Therefore, total trainable parameters $= 12 + 8 + 3 = 23$

* For biases, we use standard notation of b_{ij}, where i denotes layer number and j denotes the position of neuron in that layer. For Eg: In layer L_1, the top neuron will have bias denoted as b_{11} as it belongs to layer 1 and its position is first. Similarly, the neuron in layer 3, will have bias denoted as b_{31}.
* For output, we again use standard notation of O_{ij}, where i denotes the layer number and j denotes the position of that specific neuron in that layer. For Eg: In layer L_1, the top neuron will have outputs denotes as O_{11} & the bottom neuron will have output denoted as O_{13}.
* For weights, we use the standard notation of W_{ij}^{k}, where k denotes the layer in which the weight is input, i denotes the node/neuron position in current layer and j denotes the node/neuron position in ~~which~~ the input layer.

For Eg: For the top neuron in layer 1, it outputs O_{11} and connects to neuron in position 1 of layer 2, thus weight for that connection will be W_{11}^{2}. Similarly, In layer 2, the output of top neuron is O_{21} and it is input to the neuron in layer 3, the connection will have weight denoted as W_{11}^{3}, since input is in layer 3, position of output in last layer is 1 and input is to 1.

Notation for Forward Propogation

Layer #1

$$\begin{bmatrix} W_{11}^1 & W_{12}^1 & W_{13}^1 \\ W_{21}^1 & W_{22}^1 & W_{23}^1 \\ W_{31}^1 & W_{32}^1 & W_{33}^1 \end{bmatrix}^T_{3\times3} \begin{bmatrix} X_{i1} \\ X_{i2} \\ X_{i3} \end{bmatrix}_{3\times1} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}_{3\times1}$$

$$\Rightarrow \begin{bmatrix} W_{11}^1 & W_{21}^1 & W_{31}^1 \\ W_{12}^1 & W_{22}^1 & W_{32}^1 \\ W_{13}^1 & W_{23}^1 & W_{33}^1 \end{bmatrix}_{3\times3} \begin{bmatrix} X_{i1} \\ X_{i2} \\ X_{i3} \end{bmatrix}_{3\times1} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}_{3\times1}$$

$$\Rightarrow \begin{bmatrix} W^1_{11}X_{i1} + W^1_{21}X_{i2} + W^1_{31}X_{i3} \\ W^1_{12}X_{i1} + W^1_{22}X_{i2} + W^1_{32}X_{i3} \\ W^1_{13}X_{i1} + W^1_{23}X_{i2} + W^1_{33}X_{i3} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} \Rightarrow$$

$$\Rightarrow \underset{\text{sigmoid activation}}{\sigma}\left(\begin{bmatrix} W^1_{11}X_{i1} + W^1_{21}X_{i2} + W^1_{31}X_{i3} + b_{11} \\ W^1_{12}X_{i1} + W^1_{22}X_{i2} + W^1_{32}X_{i3} + b_{12} \\ W^1_{13}X_{i1} + W^1_{23}X_{i2} + W^1_{33}X_{i3} + b_{13} \end{bmatrix}\right) \Rightarrow \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix}_{3\times 1}$$

Layer #2

$$\begin{bmatrix} W^2_{11} & W^2_{12} \\ W^2_{21} & W^2_{22} \\ W^2_{31} & W^2_{32} \end{bmatrix}^T_{3\times 2} \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix}_{3\times 1} + \begin{bmatrix} b_{21} \\ b_{22} \\ \cancel{b} \end{bmatrix}_{2\times 1}$$

$$\Rightarrow \begin{bmatrix} W^2_{11} & W^2_{21} & W^2_{31} \\ W^2_{12} & W^2_{22} & W^2_{32} \end{bmatrix}_{2\times 3} \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix}_{3\times 1} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}_{2\times 1}$$

$$\Rightarrow \begin{bmatrix} W^2_{11}O_{11} + W^2_{21}O_{12} + W^2_{31}O_{13} \\ W^2_{12}O_{11} + W^2_{22}O_{12} + W^2_{32}O_{13} \end{bmatrix}_{2\times 1} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

$$\Rightarrow \sigma\left(\begin{bmatrix} W^2_{11}O_{11} + W^2_{21}O_{12} + W^2_{31}O_{13} \\ W^2_{12}O_{11} + W^2_{22}O_{12} + W^2_{32}O_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}\right) \Rightarrow \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix}_{2\times 1}$$

Layer #3

$$\begin{bmatrix} W^3_{11} \\ W^3_{21} \end{bmatrix}^T_{2\times 1} \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix}_{2\times 1} + \begin{bmatrix} b_{31} \end{bmatrix}_{1\times 1}$$

$$\Rightarrow \begin{bmatrix} W^3_{11} & W^3_{21} \end{bmatrix}_{1\times 2} \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix}_{2\times 1} + \begin{bmatrix} b_{31} \end{bmatrix}_{1\times 1}$$

$$\Rightarrow \begin{bmatrix} W^3_{11}O_{21} + W^3_{21}O_{22} + b_{31} \end{bmatrix}$$

$$\Rightarrow \sigma([W^3_{11}O_{21} + W^3_{21}O_{22} + b_{31}) \Rightarrow \begin{bmatrix} O_{31} \end{bmatrix}_{1\times 1} = \hat{Y}_i$$