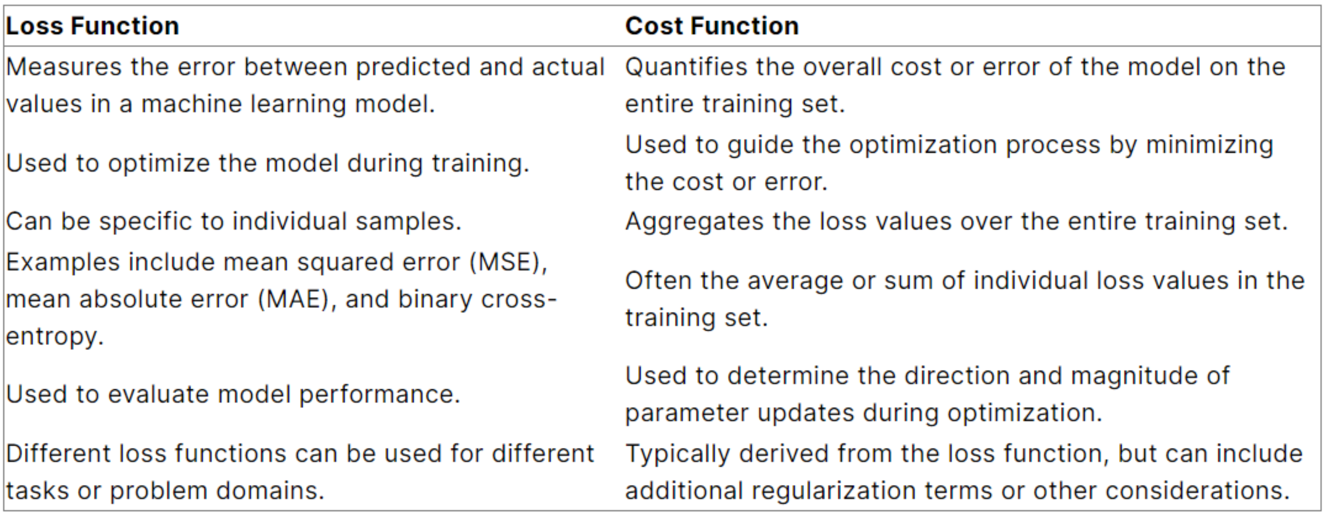
Loss Function

- A loss function is a method of evaluating how well your algorithm models your dataset.
- the function which is used to evaluate a solution is referred to as the **objective function.**
- Now we may want to maximize or minimize the objective function so to get the highest or lowest score respectively.
- deep learning neural network, we want to minimize the error value and hence the objective function here is known as a **cost function** or a **loss function** and the value of this objection function is simply referred as the **"loss".**
- the cost function and the loss function are synonymous and used interchangeably but they are a little bit different actually.
- When we have a single training example then it is known as loss function. It is also sometimes called as an error function. On the other hand, a cost function is the average loss over the entire training dataset.

Why Loss Function in Deep Learning is Important?

- Famous author Peter Druker says You can't improve what you can't measure. That's why the loss function comes into the picture to evaluate how well your algorithm is modeling your dataset.
- if the value of the loss function is lower then it's a good model otherwise, we have to change the parameter of the model and minimize the loss.

Cost Function vs Loss Function



Loss Function	**Cost Function**
Measures the error between predicted and actual values in a machine learning model.	Quantifies the overall cost or error of the model on the entire training set.
Used to optimize the model during training.	Used to guide the optimization process by minimizing the cost or error.
Can be specific to individual samples.	Aggregates the loss values over the entire training set.
Examples include mean squared error (MSE), mean absolute error (MAE), and binary cross-entropy.	Often the average or sum of individual loss values in the training set.
Used to evaluate model performance.	Used to determine the direction and magnitude of parameter updates during optimization.
Different loss functions can be used for different tasks or problem domains.	Typically derived from the loss function, but can include additional regularization terms or other considerations.

Types of loss functions

There are mainly three types of loss functions we have and again these loss functions are further divided which is shown as below

1. Regression Loss Functions

- Mean Squared Error Loss
- Mean Squared Logarithmic Error Loss
- Mean Absolute Error Loss
- L1 Loss
- L2 Loss
- Huber Loss
- Pseudo Huber Loss

2. Binary Classification Loss Functions

- Binary Cross-Entropy
- Hinge Loss
- Squared Hinge Loss

3. Multi-class Classification Loss Functions

- Multi-class Cross Entropy Loss
- Sparse Multiclass Cross-Entropy Loss
- Kullback Leibler Divergence Loss

A. Regression Loss

1. Mean Squared Error/Squared loss/ L2 loss

The Mean Squared Error (MSE) is the simplest and most common loss function. To calculate the MSE, you take the difference between the actual value and model prediction, square it, and average it across the whole dataset.

$$\text{MSE} = \frac{1}{N}\sum_{i}^{N}(Yi - \hat{Y}i)^2$$

Advantage

- 1. Easy to interpret.
- 2. Always differential because of the square.
- 3. Only one local minima.

Disadvantage

- 1. Error unit in the square. because the unit in the square is not understood properly.
- 2. Not robust to outlier

Note – In regression at the last neuron use linear activation function.

2. Mean Absolute Error/ L1 loss

The Mean Absolute Error (MAE) is also the simplest loss function. To calculate the MAE, you take the difference between the actual value and model prediction and average it across the whole dataset.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |Yi - \hat{Y}i|$$

Advantage

- 1. Intuitive and easy
- 2. Error Unit Same as the output column.
- 3. Robust to outlier

Disadvantage

- 1. Graph, not differential. we can not use gradient descent directly, then we can subgradient calculation.

Note – In regression at the last neuron use linear activation function.

3. Huber Loss

In statistics, the Huber loss is a loss function used in robust regression, that is less sensitive to outliers in data than the squared error loss.

$$Huber = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \hat{y}_i)^2 \qquad |y_i - \hat{y}_i| \leq \delta$$

$$Huber = \frac{1}{n}\sum_{i=1}^{n}\delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right) \qquad |y_i - \hat{y}_i| > \delta$$

- n – the number of data points.
- y – the actual value of the data point. Also known as true value.
- ŷ – the predicted value of the data point. This value is returned by the model.
- δ – defines the point where the Huber loss function transitions from a quadratic to linear.

Advantage

- Robust to outlier
- It lies between MAE and MSE.

Disadvantage

- Its main disadvantage is the associated complexity. In order to maximize model accuracy, the hyperparameter δ will also need to be optimized which increases the training requirements.

B. Classification Loss

1. Binary Cross Entropy/log loss

It is used in binary classification problems like two classes. example a person has covid or not or my article gets popular or not.

Binary cross entropy compares each of the predicted probabilities to the actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value.

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$$

- yi – actual values
- yihat – Neural Network prediction

Advantage –

- A cost function is a differential.

Disadvantage –

- Multiple local minima
- Not intuitive

Note – In classification at last neuron use sigmoid activation function.

2. Categorical Cross Entropy

Categorical Cross entropy is used for Multiclass classification and softmax regression.

loss function = -sum up to k(yjlagyjhat) where k is classes

$$\text{Loss} = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

where k is number of classes in the data

cost function = -1/n(sum upto n(sum j to k (yijloghijhat))

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}[y_{ij}\log(\hat{y}_{ij})]$$

where

- k is classes,
- y = actual value
- yhat – Neural Network prediction

Note – In multi-class classification at the last neuron use the softmax activation function.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

if problem statement have 3 classes

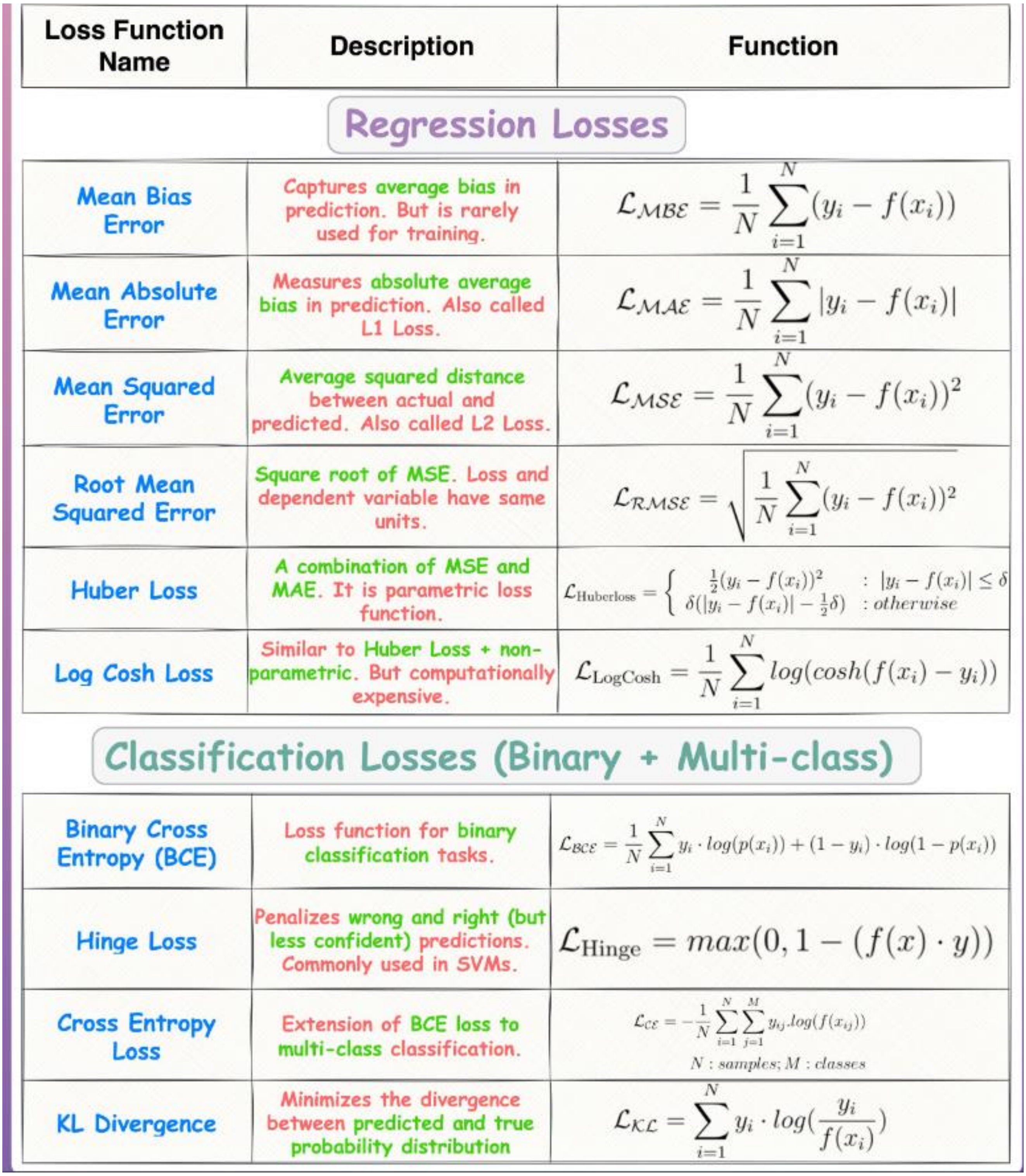
softmax activation – f(z) = ez1/(ez1+ez2+ez3)

When to use categorical cross-entropy and sparse categorical cross-entropy?

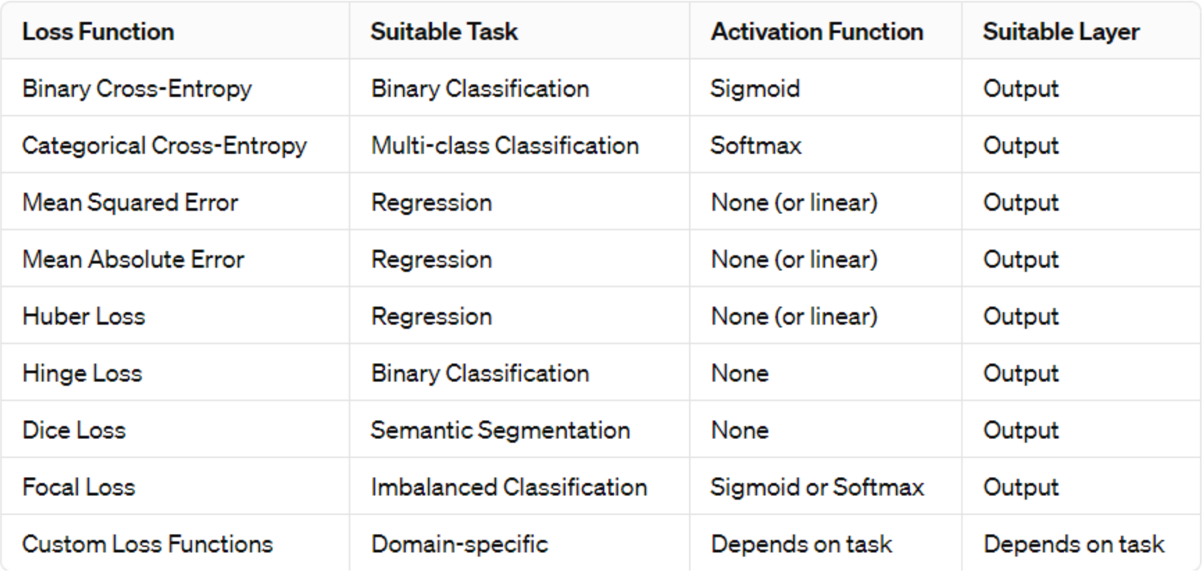
If target column has One hot encode to classes like 0 0 1, 0 1 0, 1 0 0 then use categorical cross-entropy. and if the target column has Numerical encoding to classes like 1,2,3,4….n then use sparse categorical cross-entropy.

Which is Faster?

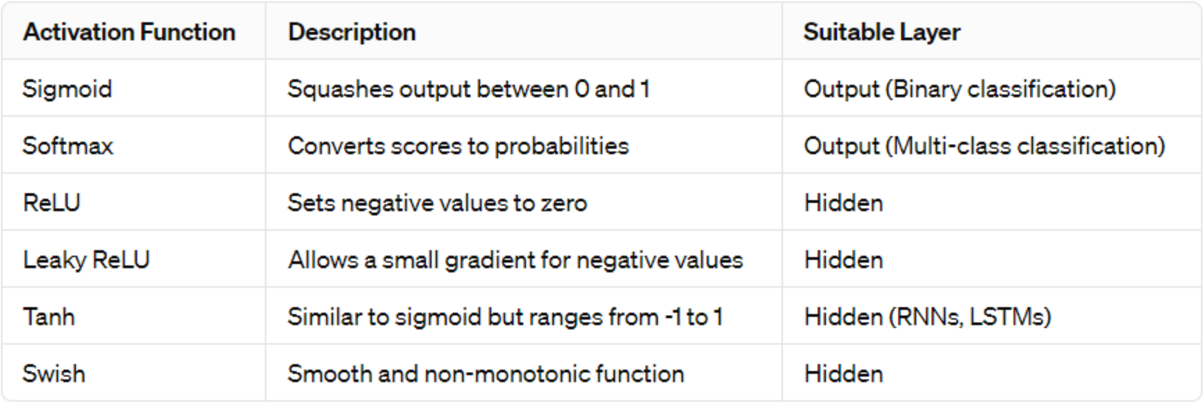
sparse categorical cross-entropy faster than categorical cross-entropy.



Loss Function Name	Description	Function
Regression Losses		
Mean Bias Error	Captures average bias in prediction. But is rarely used for training.	$\mathcal{L}_{\mathcal{MBE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))$
Mean Absolute Error	Measures absolute average bias in prediction. Also called L1 Loss.	$\mathcal{L}_{\mathcal{MAE}} = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - f(x_i)\rvert$
Mean Squared Error	Average squared distance between actual and predicted. Also called L2 Loss.	$\mathcal{L}_{\mathcal{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$
Root Mean Squared Error	Square root of MSE. Loss and dependent variable have same units.	$\mathcal{L}_{\mathcal{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$
Huber Loss	A combination of MSE and MAE. It is parametric loss function.	$\mathcal{L}_{\text{Huberloss}} = \begin{cases} \frac{1}{2}(y_i - f(x_i))^2 & : \lvert y_i - f(x_i)\rvert \leq \delta \\ \delta(\lvert y_i - f(x_i)\rvert - \frac{1}{2}\delta) & : otherwise \end{cases}$
Log Cosh Loss	Similar to Huber Loss + non-parametric. But computationally expensive.	$\mathcal{L}_{\text{LogCosh}} = \frac{1}{N}\sum_{i=1}^{N} log(cosh(f(x_i) - y_i))$
Classification Losses (Binary + Multi-class)		
Binary Cross Entropy (BCE)	Loss function for binary classification tasks.	$\mathcal{L}_{\mathcal{BCE}} = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(x_i)) + (1 - y_i) \cdot log(1 - p(x_i))$
Hinge Loss	Penalizes wrong and right (but less confident) predictions. Commonly used in SVMs.	$\mathcal{L}_{\text{Hinge}} = max(0, 1 - (f(x) \cdot y))$
Cross Entropy Loss	Extension of BCE loss to multi-class classification.	$\mathcal{L}_{\mathcal{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}.log(f(x_{ij}))$ $N : samples; M : classes$
KL Divergence	Minimizes the divergence between predicted and true probability distribution	$\mathcal{L}_{\mathcal{KL}} = \sum_{i=1}^{N} y_i \cdot log(\frac{y_i}{f(x_i)})$



Loss Function	Suitable Task	Activation Function	Suitable Layer
Binary Cross-Entropy	Binary Classification	Sigmoid	Output
Categorical Cross-Entropy	Multi-class Classification	Softmax	Output
Mean Squared Error	Regression	None (or linear)	Output
Mean Absolute Error	Regression	None (or linear)	Output
Huber Loss	Regression	None (or linear)	Output
Hinge Loss	Binary Classification	None	Output
Dice Loss	Semantic Segmentation	None	Output
Focal Loss	Imbalanced Classification	Sigmoid or Softmax	Output
Custom Loss Functions	Domain-specific	Depends on task	Depends on task



Activation Function	Description	Suitable Layer
Sigmoid	Squashes output between 0 and 1	Output (Binary classification)
Softmax	Converts scores to probabilities	Output (Multi-class classification)
ReLU	Sets negative values to zero	Hidden
Leaky ReLU	Allows a small gradient for negative values	Hidden
Tanh	Similar to sigmoid but ranges from -1 to 1	Hidden (RNNs, LSTMs)
Swish	Smooth and non-monotonic function	Hidden