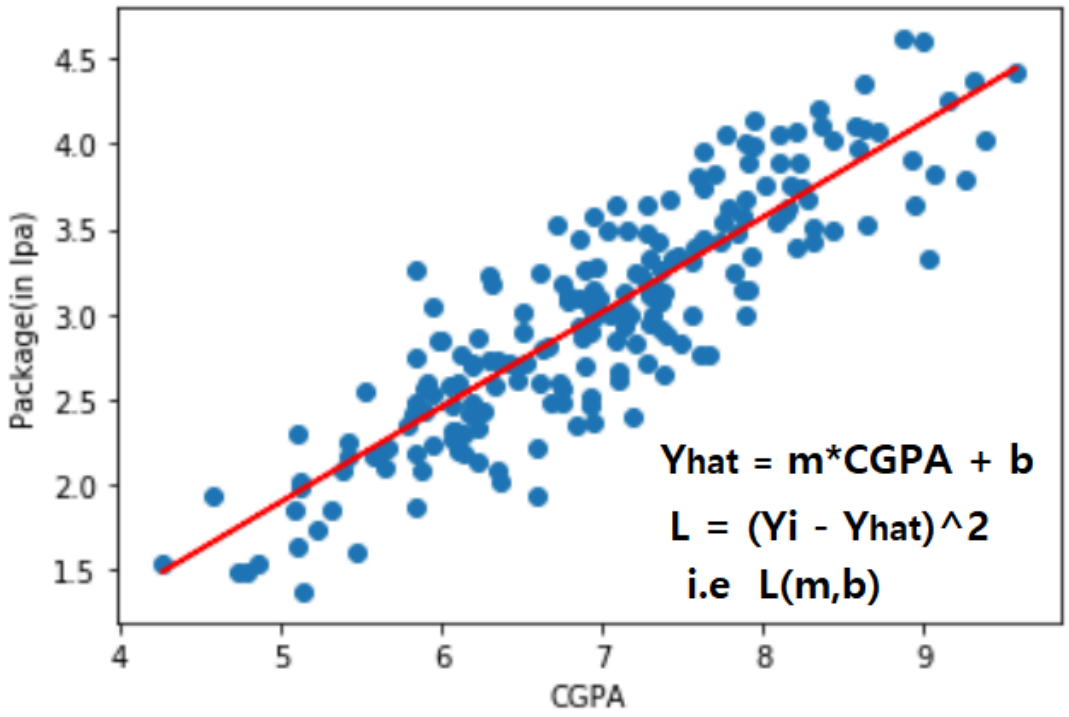
Deep learning loss functions

In mathematical optimization and decision theory, a loss or cost function (sometimes also called an error function) is a function that maps an event or values of one or more variables onto a real number intuitively representing some “cost” associated with the event.

In simple terms, the Loss function is a method of evaluating how well your algorithm is modeling your dataset. It is a mathematical function of the parameters of the machine learning algorithm.

In simple linear regression, prediction is calculated using slope(m) and intercept(b). the loss function for this is the (Yi – Yihat)^2 i.e loss function is the function of slope and intercept.



Why Loss Function in Deep Learning is Important?

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 in Deep Learning

Most people confuse loss function and cost function. let's understand what is loss function and cost function. Cost function and Loss function are synonymous and used interchangeably but they are different.

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.

Loss Function in Deep Learning

Regression

- MSE(Mean Squared Error)
- MAE(Mean Absolute Error)
- Hubber loss

Classification

- Binary cross-entropy
- Categorical cross-entropy

AutoEncoder

- KL Divergence

GAN

- Discriminator loss
- Minmax GAN loss

Object detection

- Focal loss

Word embeddings

- Triplet loss

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.

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

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.

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.

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

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
- Y hat – 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.

- We learned the importance of loss function in deep learning.
- Difference between loss and cost.
- The mean absolute error is robust to the outlier.
- This function is used for binary classification.
- Sparse categorical cross-entropy is faster than categorical cross-entropy.

Autoencoder:

An autoencoder is a type of artificial neural network used for unsupervised learning. It aims to learn efficient representations of data, typically by reducing the dimensionality of the input data. The autoencoder consists of an encoder network that maps the input data into a lower-dimensional latent space representation, and a decoder network that reconstructs the input data from the latent space representation. During training, the autoencoder learns to minimize the reconstruction error, typically measured using a loss function like mean squared error (MSE). Autoencoders are widely used for tasks such as data compression, denoising, and feature learning.

KL Divergence (Kullback-Leibler Divergence):

KL divergence is a measure of how one probability distribution diverges from a second, expected probability distribution. In the context of autoencoders, the KL divergence is often used as part of the loss function in variational autoencoders (VAEs). VAEs are a type of generative model that combines the structure of an autoencoder with probabilistic latent variables. The KL divergence term in the VAE loss function encourages the learned latent space to approximate a predefined probability distribution, typically a Gaussian distribution. This regularization term ensures that the

latent space remains smooth and continuous, which helps generate high-quality samples during the decoding process.

Kullback-Leibler Divergence

The KL divergence score, quantifies how much one probability distribution differs from another probability distribution.

The KL divergence between two distributions Q and P is often stated using the following notation:

- KL(P || Q)

Where the "||" operator indicates "*divergence*" or Ps divergence from Q.

KL divergence can be calculated as the negative sum of probability of each event in P multiplied by the log of the probability of the event in Q over the probability of the event in P.

- KL(P || Q) = – sum x in X P(x) * log(Q(x) / P(x))

The value within the sum is the divergence for a given event.

This is the same as the positive sum of probability of each event in P multiplied by the log of the probability of the event in P over the probability of the event in Q (e.g. the terms in the fraction are flipped). This is the more common implementation used in practice.

- KL(P || Q) = sum x in X P(x) * log(P(x) / Q(x))

Generative Adversarial Network (GAN)

Discriminator Loss:

- In the context of GANs, the discriminator's loss refers to the loss function used to train the discriminator network.
- The goal of the discriminator is to classify whether a given input is real (coming from the true data distribution) or fake (generated by the generator).
- The discriminator's loss is typically a binary cross-entropy loss.
- It aims to correctly classify real samples as real (label 1) and fake samples as fake (label 0).
- The discriminator's loss is minimized during training to improve its ability to distinguish between real and fake samples.

Minimax GAN Loss:

- The minimax loss, often referred to as the adversarial loss, is the primary objective function used in GAN training.

- It reflects the competitive nature of the game between the generator and the discriminator.
- The minimax loss is formulated as a minimization-maximization problem.
- The objective is to find the Nash equilibrium where the generator produces realistic samples that fool the discriminator, while the discriminator becomes unable to distinguish between real and fake samples.

Focal Loss:

A Focal Loss function addresses class imbalance during training in tasks like object detection. Focal loss applies a modulating term to the cross entropy loss in order to focus learning on hard misclassified examples. It is a dynamically scaled cross entropy loss, where the scaling factor decays to zero as confidence in the correct class increases. Intuitively, this scaling factor can automatically down-weight the contribution of easy examples during training and rapidly focus the model on hard examples.The focal loss formula is given below

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

where:

p_t is the predicted probability of the true class.

α_t is a weighting factor that balances the importance of positive and negative examples.

γ is a focusing parameter that modulates the rate at which the loss decreases as p_t *increases.*

Focal loss has been shown to be effective in improving the performance of object detection models, particularly in scenarios with severe class imbalance and difficult examples. It is commonly used in conjunction with convolutional neural networks for various object detection tasks.

Word Embeddings:

Word embeddings are dense vector representations of words in a continuous vector space, typically in a high-dimensional space. In natural language processing (NLP), word embeddings are used to capture semantic similarities between words based on their context in a corpus of text. The underlying idea is that words with similar meanings should have similar vector representations, and

these representations can be learned from large text corpora using techniques like Word2Vec, GloVe (Global Vectors for Word Representation), or fastText.

Triplet Loss:

Triplet loss is a loss function used in siamese neural networks and related architectures for learning embeddings. Siamese networks are neural networks that share the same architecture and weights, used to learn embeddings of input data such that similar inputs are mapped closer together in the embedding space.

In the context of triplet loss, each training sample consists of three inputs: an anchor input, a positive input (similar to the anchor), and a negative input (dissimilar to the anchor). The goal of triplet loss is to encourage the embeddings of the anchor and positive examples to be closer in the embedding space while pushing the embedding of the negative example farther away.

The triplet loss function is typically defined as:

$$L=\max(0,d(A,P)-d(A,N)+\alpha)$$

where:

- $d(A,P)$ is the distance between the embeddings of the anchor and positive examples,
- $d(A,N)$ is the distance between the embeddings of the anchor and negative examples,
- α is a margin hyperparameter that controls the minimum desired distance between the anchor-positive pair and the anchor-negative pair,
- L is the triplet loss, which is minimized during training.

The objective of triplet loss is to learn embeddings such that the distance between the anchor and positive example is smaller than the distance between the anchor and negative example by at least the margin

α. This encourages the embeddings to capture the inherent similarities and differences between the input samples.

Triplet loss is commonly used in tasks such as face recognition, image retrieval, and learning semantic embeddings in NLP, where learning embeddings that capture relationships between data points is crucial.