Loss Functions

What is a Loss Function?

- Loss functions have an important role in neural network learning as they guide the learning process of the model.
- There is a large number of loss functions available and choosing the proper one is crucial for training an accurate model. Different choices of a loss function can lead to different classification or regression models.
- In classification or regression training phage, the goal is to build a model whose predictions are as close as possible to the true labels.

- A loss function measures the model's prediction error for a given sample, i.e., the difference between the model's predicted value and the true value for that sample.
- It takes two parameters: the true label of the sample y and the model's prediction ŷ:

$$\mathcal{L}(y, \hat{y}) \text{ or } \mathcal{J}(y, \hat{y})$$

- During the training of the model, we tune its parameters so as to minimize the loss function on the given training samples.
- Loss function is also known as cost function, in some cases it computes instantaneous error, otherwise calculates the error over the whole dataset.

Desired Properties of a Loss Function

- Ideally, we would like the loss function to have the following properties:
- The loss function should reflect the objective the model is trying to achieve. For example, in regression problems our goal is to minimize the differences between the predictions and the target values, while in classification our goal is to minimize the number of misclassification errors.
- Continuous and differentiable everywhere. Most optimization algorithms, such as gradient descent, require the loss function to be differentiable.

- Convex. A convex function has only one global minimum point, thus optimization methods like gradient descent are guaranteed to return the globally optimal solution. In practice, this property is hard to achieve, and most loss functions are non-convex (i.e., they have multiple local minima).
- Symmetric, i.e., the error above the target should cause the same loss as the same error below the target.
- Fast to compute.

Loss function for Binary Classification Problems

- In binary classification problems, the ground truth labels are binary (1/0 or 1/-1).
- The predicted value of the model can be either binary (a hard label) or a probability estimate that the given sample belongs to the positive class (a soft label).
- 0–1 Loss:The simplest loss function is the zero-one loss function (also called misclassification error).

$$L_{01}(y, \hat{y}) = I(y \neq \hat{y})$$

Zero-one loss

- I is the indicator function that returns 1 if its input is true, and 0 otherwise.

- For every sample that the classifier gets wrong (misclassifies) a loss of 1 is incurred, whereas correctly classified samples lead to 0 loss.
- The 0–1 loss function is often used to evaluate classifiers, but is not useful in guiding optimization since it is non-differentiable and non-continuous.
- **Binary cross-entropy loss (Log loss):**It is used to train models that provide class probability estimates.
- Let us denote the probability estimate given by the model

$$p = P(y = 1)$$

- **Then log loss is defined as:**

$$L_{\log}(y, p) = -y \log p - (1 - y) \log(1 - p)$$

- The log loss function is differentiable and convex, i.e., it has a unique global minimum.
- **Multi-Class Classification Problems:**
- The loss function used to train a multi-class classifier is called **cross-entropy loss**, which is an extension of **log loss** to the multi-class case. It is defined as follows:

$$L_{\mathrm{CE}}(\mathbf{y}, \mathbf{p}) = -\sum_{i=1}^{k} y_i \log p_i$$

- For example, assume that we have a three-class problem, the true class of our sample is class 2 (i.e., y = [0, 1, 0]), and the prediction of our model is p = [0.3, 0.6, 0.1].

Continued (Cross Entropy Loss)

- Then the cross-entropy loss induced by this sample is:

$$L_{\mathrm{CE}} = -(0 \cdot \log 0.3 + 1 \cdot \log 0.6 + 0 \cdot \log 0.1) = 0.5108$$

- To see how the cross-entropy loss generalizes log loss, notice that in the binary case $p_1 = 1 - p_0$ and $y_1 = 1 - y_0$, therefore we get:

$$L_{\mathrm{BinaryCE}}(\mathbf{y}, \mathbf{p}) = -y_0 \log p_0 - y_1 \log p_1 = -y_0 \log p_0 - (1 - y_0) \log(1 - p_0)$$

- In multi-class classification problems, cross-entropy loss is the most common loss function and is extension of log loss to the multi-class case.

Mean Squared Error Loss Function

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2$$

y_i : entries in the prediction vector $\vec{y}$

$\hat{y}_i$: entries in the ground truth label $\hat{\vec{y}}$

- You divide the sum of squared differences by N, which corresponds to the length of the vectors. If the output y of your neural network is a vector with multiple entries then N is the number of the vector entries with y_i being one particular entry in the output vector.
- The mean squared error loss function is the perfect loss function if you're dealing with a regression problem. That is, if you want your neural network to predict a **continuous scalar value**.