# Deep Learning

# Cross-Entropy Loss Function

**Sachchida Nand Chaurasia**

Assistant Professor

Department of Computer Science

Banaras Hindu University

Varanasi

Email id: snchaurasia@bhu.ac.in, sachchidanand.mca07@gmail.com

July 31, 2023

## Cross-Entropy Loss Function I

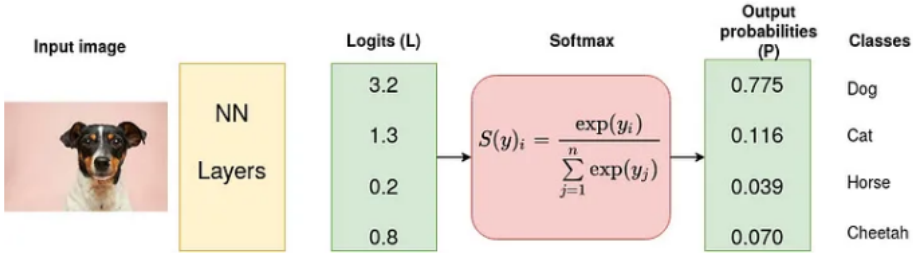
✓ When working on a Machine Learning or a Deep Learning Problem, loss/cost functions are used to optimize the model during training.

✓ The objective is almost always to minimize the loss function. The lower the loss the better the model.

✓ Cross-Entropy loss is a most important cost function. It is used to optimize classification models. The understanding of Cross-Entropy is pegged on understanding of Softmax activation function.

✓ Consider a 4-class classification task where an image is classified as either a dog, cat, horse or cheetah.

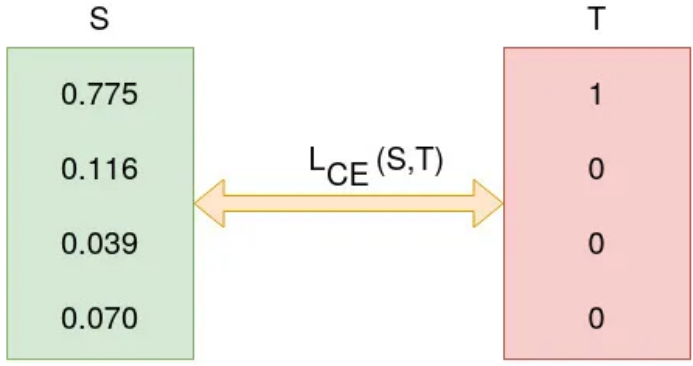
# Cross-Entropy Loss Function II




Input image source: Photo by Victor Grabarczyk on Unsplash . Diagram by author.

✓ In the above Figure, Softmax converts logits into probabilities. The purpose of the Cross-Entropy is to take the output probabilities (P) and measure the distance from the truth values (as shown in Figure below).

# Cross-Entropy Loss Function III



## Cross-Entropy Loss Function IV

✓ For the example above the desired output is [1,0,0,0] for the class dog but the model outputs [0.775, 0.116, 0.039, 0.070] .

✓ The objective is to make the model output be as close as possible to the desired output (truth values).

✓ During model training, the model weights are iteratively adjusted accordingly with the aim of minimizing the Cross-Entropy loss. The process of adjusting the weights is what defines model training and as the model keeps training and the loss is getting minimized, we say that the model is learning.

## Entropy

✓ Entropy of a random variable X is the level of uncertainty inherent in the variables possible outcome.

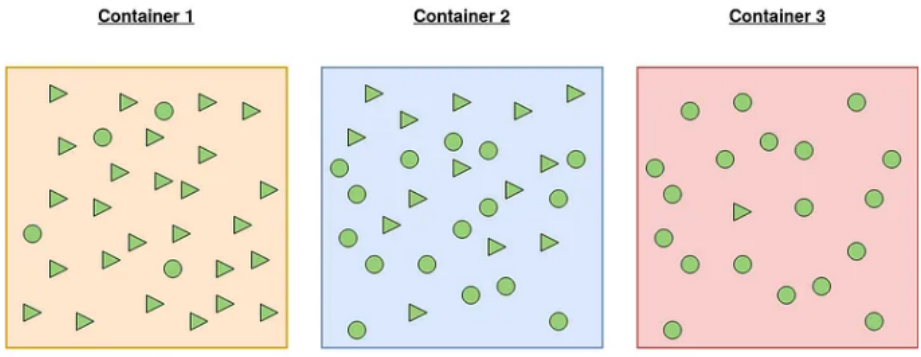
✓ For $p(x)$ — probability distribution and a random variable X, entropy is defined as follows:

$$H(X) = \begin{cases} -\int_x p(x) \log p(x) & \text{X is continous} \\ \sum_x p(x) \log p(x) & \text{X is discrete} \end{cases}$$

The greater the value of entropy, $H(x)$ , the greater the uncertainty for probability distribution and the smaller the value the less the uncertainty.

# Cross-Entropy Loss Function VI

## Example




3 containers with triangle and circle shapes. (Source: Author).

## Cross-Entropy Loss Function VII

✓ *Container 1: The probability of picking a triangle is 26/30 and the probability of picking a circle is 4/30. For this reason, the probability of picking one shape and/or not picking another is more certain.*

✓ *Container 2: Probability of picking the a triangular shape is 14/30 and 16/30 otherwise. There is almost 50–50 chance of picking any particular shape. Less certainty of picking a given shape than in 1.*

✓ *Container 3: A shape picked from container 3 is highly likely to be a circle. Probability of picking a circle is 29/30 and the probability of picking a triangle is 1/30. It is highly certain than the shape picked will be circle.*

Let us calculate the entropy so that we ascertain our assertions about the certainty of picking a given shape.

## Cross-Entropy Loss Function VIII

**Entropy for container 1:**

$$
\begin{aligned}
H(X) &= -\sum_{x} p(x) \log(p(x)) \\
&= -\left[p(x_1) \log_2 \left(p(x_1)\right) + p(x_2) \log_2 \left(p(x_2)\right)\right] \\
&= -\left[\frac{26}{30} \log_2 \left(\frac{26}{30}\right) + \frac{4}{30} \log_2 \left(\frac{4}{30}\right)\right] \\
&= 0.5665
\end{aligned}
$$

## Cross-Entropy Loss Function IX

**Entropy for container 2:**

$$
\begin{aligned}
H(X) &= -\sum_{x} p(x) \log(p(x)) \\
&= -\left[p(x_1) \log_2 \left(p(x_1)\right) + p(x_2) \log_2 \left(p(x_2)\right)\right] \\
&= -\left[\frac{14}{30} \log_2 \left(\frac{14}{30}\right) + \frac{16}{30} \log_2 \left(\frac{16}{30}\right)\right] \\
&= 0.9968
\end{aligned}
$$

**Entropy for container 3:**

$$
\begin{aligned}
H(X) &= -\sum_{x} p(x) \log(p(x)) \\
&= -\left[p(x_1) \log_2 \left(p(x_1)\right) + p(x_2) \log_2 \left(p(x_2)\right)\right] \\
&= -\left[\frac{1}{30} \log_2 \left(\frac{1}{30}\right) + \frac{29}{30} \log_2 \left(\frac{29}{30}\right)\right] \\
&= 0.2108
\end{aligned}
$$

✓ As expected the entropy for the first and third container is smaller than the second one. This is because probability of picking a given shape is more certain in container 1 and 3 than in 2.

## Cross-Entropy Loss Function (logarithmic loss, log loss or logistic loss):

✓ Each predicted class probability is compared to the actual class desired output 0 or 1 and a score/loss is calculated that penalizes the probability based on how far it is from the actual expected value.

✓ The penalty is logarithmic in nature yielding a large score for large differences close to 1 and small score for small differences tending to 0.

✓ Cross-entropy loss is used when adjusting model weights during training. The aim is to minimize the loss, i.e, the smaller the loss the better the model. A perfect model has a cross-entropy loss of 0.

## Cross-entropy is defined as:

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i)$$

where $t_i$ is the truth label and $p_i$ is the Softmax probability for the $i^{th}$ class. $n$ is numbe of classes.

## Binary Cross-Entropy Loss

✓ For binary classification (a classification task with two classes — 0 and 1), we have binary cross-entropy defined as:

$$\begin{aligned} L &= -\sum_{i=1}^{2} t_i \log(p(i) \\ &= -\left[t_1 \log(p_1) + t_2 \log(p_2)\right] \\ &= -\left[t \log(p) + (1-t) \log(1-p)\right] \end{aligned}$$

where $t_i$ is the truth value taking a value 0 or 1 and $p_i$ is the Softmax probability for the $i^{th}$ class. Since we have two classess 1 and 0 we can have $t_1 = 1$ and $t_2 = 0$ and since $p'$s are

## Cross-Entropy Loss Function XIV

probabilities then $p_1 + p_2 = 1 \longrightarrow p_1 = 1 - p_2$. For the convenience of notation, we can then let $t_1 = t$, $t_2 = 1 - t$, $p_1 = p$ and $p_2 = 1 - p$.

✓ Binary cross-entropy is often calculated as the average cross-entropy across all data examples, that is,

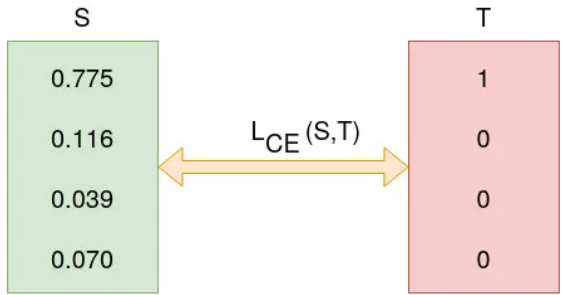
$$L = -\frac{1}{N}\left[\sum_{j=1}^{N}[t_j \log(p_j) + (1 - t_j)\log(1 - p_j)]\right]$$

for $N$ data points where $t_i$ is the truth value taking a value 0 or 1 and $p_i$ is the Softmax probability for the $i^{th}$ data point.

## Cross-Entropy Loss Function XV

**Example:**

Consider the classification problem with the following Softmax probabilities (S) and the labels (T). The objective is to calculate for cross-entropy loss given these information.




Logits(S) and one-hot encoded truth label(T) with Categorical Cross-Entropy loss function used to measure the 'distance' between the predicted probabilities and the truth labels. (Source: Author)

## Cross-Entropy Loss Function XVI

The categorical cross-entropy is computed as follows:

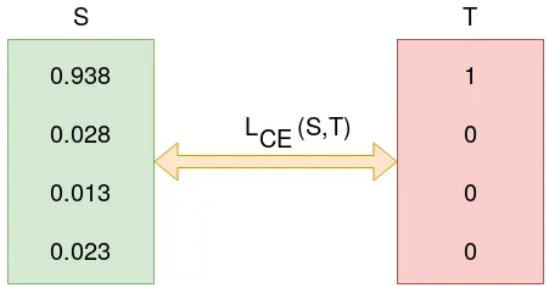
$$
\begin{aligned}
L_{CE} &= -\sum_{i=1} T_i \log(S_i) \\
&= -\left[1 * \log_2(0.775) + 0 * \log_2(0.126) + 0 * \log_2(0.039) + 0 * \log_2(0.070)\right] \\
&= -\log(0.775) \\
&= 0.3677
\end{aligned}
$$

✓ Softmax is continuously differentiable function. This makes it possible to calculate the derivative of the loss function with respect to every weight in the neural network.

## Cross-Entropy Loss Function XVII

✓ This property allows the model to adjust the weights accordingly to minimize the loss function (model output close to the true values).

✓ Assume that after some iterations of model training the model outputs the following vector of logits



## Cross-Entropy Loss Function XVIII

$$\begin{aligned} L_{CE} &= -1 * \log_2(0.938) + 0 + 0 + 0 \\ &= 0.095 \end{aligned}$$

0.095 is less than previous loss, that is, 0.3677 implying that the model is learning. The process of optimization (adjusting weights so that the output is close to true values) continues until training is over.

## Softmax I

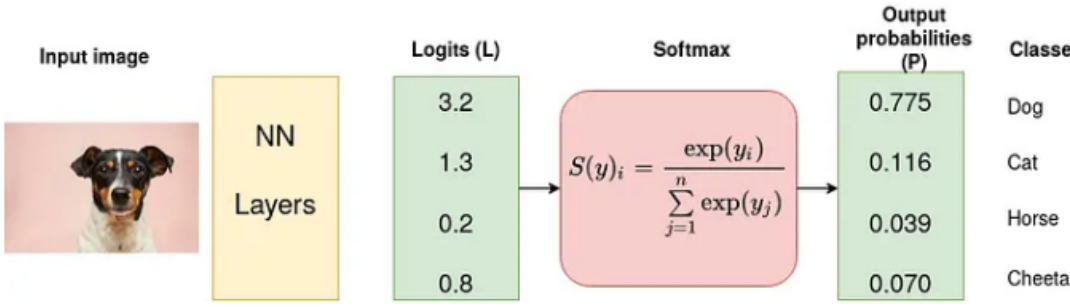
✓ It is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes.

✓ Softmax is an activation function that scales numbers/logits into probabilities.

✓ The output of a Softmax is a vector (say v) with probabilities of each possible outcome. The probabilities in vector v sums to one for all possible outcomes or classes.

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_i)}$$

# Softmax II




Input image source: Photo by Victor Grabarczyk on Unsplash . Diagram by author.

# Softmax III

$$\exp(3.2) = 24.5325$$
$$\exp(1.3) = 3.6693$$
$$\exp(0.2) = 1.2214$$
$$\exp(0.8) = 2.2255$$

and therefore,

$$\begin{aligned} S(3.2) &= \frac{\exp(3.2)}{\exp(3.2) + \exp(1.3) + \exp(0.2) + \exp(0.8)} \\ &= \frac{24.5325}{24.5325 + 3.6693 + 1.2214 + 2.2255} \\ &= 0.775 \end{aligned}$$

## Categorical Data into Numerical Data:

✓ The truth labels are categorical data: any particular image can be categorized into one of these groups: dog, cat, horse or cheetah.

✓ The computer however does not understand this kind of data and therefore we need to convert them into numerical data. There are two ways to do so:

1. Integer encoding
2. One-hot encoding

**Integer Encoding (Also called Label Encoding):**

✓ In this kind of encoding, labels are assigned unique integer values. For example in our case, we will have,

0 for “dog”, 1 for “cat”, 2 for “horse” and 3 for “cheetah”.

## Softmax V

✓ When to use integer encoding: It is used when the labels are ordinal in nature, that is, labels with some order, for example, consider a classification problem where we want to classify a service as either poor, neutral or good, then we can encode these classes as follows

0 for “poor”, 1 for “neutral” and 2 for “good”.

Clearly, the labels have some order and the labels gives the weights to the labels accordingly.

✓ Conversely, we refrain from using integer encoding when the labels are nominal (names without specific ordering), for example, consider the flower classification problem where we have 3 classes: Rose, Lotus and , Sunflower.

0 for “Rose”, 1 for “Lotus” and 2 for “Sunflower”.

## Softmax VI

✓ The model may take a natural ordering of the labels (2>1>0) and give more weight to one class over another when in fact these are just labels with no specific ordering implied.

**One-hot encoding:**

✓ For categorical variables where no such ordinal relationship exists, the integer encoding is not enough. One-hot encoding is preferred.

✓ In one-hot encoding, the labels are represented by a binary variable (1 and 0s) such that for a given class a binary variable with 1 for position corresponding to that specific class and 0 elsewhere is generated, for example, in our case we will have the following labels for our 4 classes

[1, 0, 0, 0] for “dog”, [0, 1, 0, 0] for “cat”, [0, 0, 1, 0] for “horse” and [0, 0, 0, 1] for “cheetah”.

## Softmax VII

*But one may ask, why not use standard normalization, that is, take each logit and divide it by the sum of the all logits to get the probabilities? Why take the exponents?* Here are some two reasons.

1. Softmax normalization reacts to small and large variation/change differently but standard normalization does not differentiate the stimulus by intensity so longest the proportion is the same, for example,
   # Softmax normalization
   softmax([2, 4]) = [0.119, 0.881]
   softmax([4, 8]) = [0.018, 0.982]
   # Standard normalization
   std_norm([2, 4]) = [0.333, 0.667]

## Softmax VIII

std_norm([4, 8]) = [0.333, 0.667]

2. Another problem arises when there are negative values in the logits. In that case, you will end up with negative probabilities in the output. *The Softmax is not affected with negative values because exponent of any value (positive or negative) is always a positive value*.