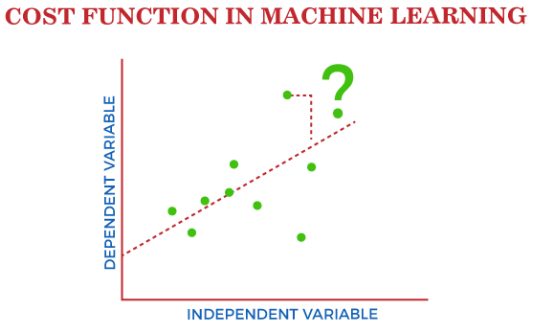
Week 06 : Cost Functions

Cost Function in Machine Learning

A Machine Learning model should have a very high level of accuracy in order to perform well with real-world applications. But how to calculate the accuracy of the model, i.e., how good or poor our model will perform in the real world? In such a case, the Cost function comes into existence. It is an important machine learning parameter to correctly estimate the model.



Cost function also plays a crucial role in understanding that how well your model estimates the relationship between the input and output parameters.

What is Cost Function?

A cost function is an important parameter that determines how well a machine learning model performs for a given dataset. It calculates the difference between the expected value and predicted value and represents it as a single real number.

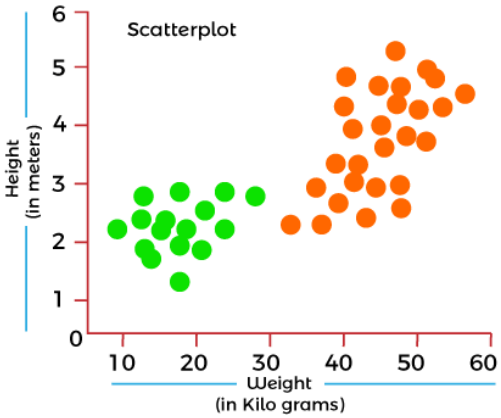
In machine learning, once we train our model, then we want to see how well our model is performing. Although there are various accuracy functions that tell you how your model is performing, but will not give insights to improve them. So, we need a function that can find when the model is most accurate by finding the spot between the undertrained and overtrained model.

In simple, "***Cost function is a measure of how wrong the model is in estimating the relationship between X(input) and Y(output) Parameter***." A cost function is sometimes also referred to as Loss function, and it can be estimated by iteratively running the model to compare estimated predictions against the known values of Y.

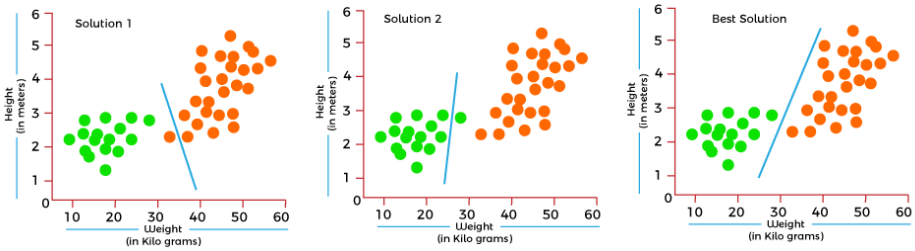
The main aim of each ML model is to determine parameters or weights that can minimize the cost function.

Why use Cost Function?

While there are different accuracy parameters, then why do we need a Cost function for the Machine learning model. So, we can understand it with an example of the classification of data. Suppose we have a dataset that contains the height and weights of cats & dogs, and we need to classify them accordingly. If we plot the records using these two features, we will get a scatter plot as below:



In the above image, the green dots are cats, and the yellow dots are dogs. Below are the three possible solutions for this classification problem.



In the above solutions, all three classifiers have high accuracy, but the third solution is the best because it correctly classifies each datapoint. The reason behind the best classification is that it is in mid between both the classes, not close or not far to any of them.
To get such results, we need a Cost function. It means for getting the optimal solution; we need a Cost function. It calculated the difference between the actual values and predicted values and measured how wrong was our model in the prediction. By minimizing the value of the cost function, we can get the optimal solution.

Gradient Descent: Minimizing the cost function
As we discussed in the above section, the cost function tells how wrong your model is? And each machine learning model tries to minimize the cost function in order to give the best results. Here comes the role of Gradient descent.
"***Gradient Descent is an optimization algorithm which is used for optimizing the cost function or error in the model.***" It enables the models to take the gradient or direction to reduce the errors by reaching to least possible error. Here direction refers to how model parameters should be corrected to further reduce the cost function. The error in your model can be different at different points, and you have to find the quickest way to minimize it, to prevent resource wastage.

Gradient descent is an iterative process where the model gradually converges towards a minimum value, and if the model iterates further than this point, it produces little or zero changes in the loss. This point is known as convergence, and at this point, the error is least, and the cost function is optimized.
Below is the equation for gradient descent in linear regression:

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta}$$

In the gradient descent equation, alpha is known as the learning rate. This parameter decides how fast you should move down to the slope. For large alpha, take big steps, and for small alpha value, you need to take small steps.
Types of Cost Function
Cost functions can be of various types depending on the problem. However, mainly it is of three types, which are as follows:

1. Regression Cost Function
2. Binary Classification cost Functions
3. Multi-class Classification Cost Function.

1. Regression Cost Function
Regression models are used to make a prediction for the continuous variables such as the price of houses, weather prediction, loan predictions, etc. When a cost function is used with Regression, it is known as the "Regression Cost Function." In this, the cost function is calculated as the error based on the distance, such as:

1. Error= Actual Output-Predicted output

There are three commonly used Regression cost functions, which are as follows:

a. Means Error
In this type of cost function, the error is calculated for each training data, and then the mean of all error values is taken. It is one of the simplest ways possible.
The errors that occurred from the training data can be either negative or positive. While finding mean, they can cancel out each other and result in the zero-mean error for the model, so it is not recommended cost function for a model. However, it provides a base for other cost functions of regression models.
b. Mean Squared Error (MSE)

Means Square error is one of the most commonly used Cost function methods. It improves the drawbacks of the Mean error cost function, as it calculates the square of the difference between the actual value and predicted value. Because of the square of the difference, it avoids any possibility of negative error.
The formula for calculating MSE is given below:

$$MSE = \sum(y - y')^2 / n$$

Mean squared error is also known as L2 Loss.
In MSE, each error is squared, and it helps in reducing a small deviation in prediction as compared to MAE. But if the dataset has outliers that generate more prediction errors, then squaring of this error will further increase the error multiple times. Hence, we can say MSE is less robust to outliers.
c. Mean Absolute Error (MAE)
Mean Absolute error also overcome the issue of the Mean error cost function by taking the absolute difference between the actual value and predicted value.

The formula for calculating Mean Absolute Error is given below:

$$\text{MAE} = \frac{\sum_{i=0}^{n} |y - y'|}{n}$$

This means the Absolute error cost function is also known as **L1 Loss**. It is not affected by noise or outliers, hence giving better results if the dataset has noise or outlier.

2. Binary Classification Cost Functions

Classification models are used to make predictions of categorical variables, such as predictions for 0 or 1, Cat or dog, etc. The cost function used in the classification problem is known as the Classification cost function. However, the classification cost function is different from the Regression cost function.

One of the commonly used loss functions for classification is cross-entropy loss.

The binary Cost function is a special case of Categorical cross-entropy, where there is only one output class. For example, classification between red and blue.

To better understand it, let's suppose there is only a single output variable Y

1. Cross-entropy(D) = - y*log(p) when y = 1

2. Cross-entropy(D) = - (1-y)*log(1-p) when y = 0

The error in binary classification is calculated as the mean of cross-entropy for all N training data. Which means:

1. Binary Cross-Entropy = (Sum of Cross-Entropy for N data)/N

3. Multi-class Classification Cost Function

A multi-class classification cost function is used in the classification problems for which instances are allocated to one of more than two classes. Here also, similar to binary class classification cost function, cross-entropy or categorical cross-entropy is commonly used cost function.

It is designed in a way that it can be used with multi-class classification with the target values ranging from 0 to 1, 3, ….,n classes.

In a multi-class classification problem, cross-entropy will generate a score that summarizes the mean difference between actual and anticipated probability distribution.

For a perfect cross-entropy, the value should be zero when the score is minimized.

4. Log Loss

Log loss, also known as logarithmic loss or cross-entropy loss, is a common evaluation metric for binary classification models. It measures the performance of a model by quantifying the difference between predicted probabilities and actual values. Log-loss is indicative of how close the prediction probability is to the corresponding actual/true value (0 or 1 in case of binary classification). Log Loss is a standard way of interpreting the cross-entropy that comes from the field of information theory. The logistic loss or cross-entropy loss (or cross entropy) is used in classification problems.

Binary Cross-Entropy = (Sum of Cross-Entropy for N data)/N

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

Yi represents the actual class
log(p(yi)) is the probability of that class
p(yi) is the probability of 1.
1-p(yi) is the probability of 0

a. Visualization of binary cross entropy function for logistic regression

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

m or n — number of training samples

i — ith training sample in a data set
y(i) — Actual value for the ith training sample
y_hat(i) — Predicted value for the ith training sample

Graph of -log(1-X) : y=0 case

$$Cost\left(y^{(i)}, \hat{y}^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} -y^{(i)} * \log\left(\hat{y}^{(i)}\right) - (1 - y^{(i)}) * \log\left(1 - \hat{y}^{(i)}\right)$$

Step — 1: The value of cost function when Yi=0 (only the second term)

$$Cost\left(y^{(i)}, \hat{y}^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} -\log\left(1 - \hat{y}^{(i)}\right)$$

Step — 2: Assume we have only one training example. It means that n=1. So, the value of the cost function when Y=0

$$Cost = -\log(1 - \hat{y})$$

Graph of -log(X) : y=1 case

$$Cost\left(y^{(i)}, \hat{y}^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} -y^{(i)} * \log\left(\hat{y}^{(i)}\right) - (1 - y^{(i)}) * \log\left(1 - \hat{y}^{(i)}\right)$$

Step — 3: The value of cost function when Yi=1 (only the first term)

$$Cost\left(y^{(i)}, \hat{y}^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} -\log\left(\hat{y}^{(i)}\right)$$

Step — 4: Assume we have only one training example. It means that n=1. So, the value of the cost function when Y=1

$$Cost = -\log(\hat{y})$$

Derivation of the update equation for logistic regression using stochastic gradient descent

consider a single training example with input features x1 and x2 and a true label y. The cost function for logistic regression is given by:

Cost(b0,b1,b2)= - y·log(prediction)−(1−y)·log(1−prediction)
= -y. log(sigmoid(z))-(1-y)*log(1-sigmoid(z))

Taking the partial derivative of the cost function with respect to b0, b1, and b2, we get

Note: sigmoid(z)= prediction, so we get (prediction–y) as derivative of cost function

These are the gradients that guide the parameter updates in the direction that reduces the cost function and improves the model's predictions. The learning rate α scales the size of these updates during each iteration of stochastic gradient descent. Update equations for stochastic gradient descent involve subtracting the gradient term from the current parameter values, in order to move in the direction of minimizing the cost function

B0 = B0 + alpha * (y - prediction) * prediction * (1 - prediction)* 1
B1 = B1 + alpha * (y - prediction) * prediction * (1 - prediction)* x1
B2 = B2 + alpha * (y - prediction) * prediction * (1 - prediction)* x 2

Note: prediction*(1−prediction): This part incorporates the sigmoid function's derivative.
(prediction–y) : Error term (which is also derivative of cost function)

Stochastic Gradient Descent Update: In stochastic gradient descent (SGD), we update the parameters using the negative gradient of the cost function

$$
\begin{aligned}
b_0 &= b_0 - \alpha \cdot \frac{\partial \text{Cost}}{\partial b_0} \\
b_1 &= b_1 - \alpha \cdot \frac{\partial \text{Cost}}{\partial b_1} \\
b_2 &= b_2 - \alpha \cdot \frac{\partial \text{Cost}}{\partial b_2}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \text{Cost}}{\partial b_0} &= \text{sigmoid}(z) - y \\
\frac{\partial \text{Cost}}{\partial b_1} &= (\text{sigmoid}(z) - y) \cdot x_1 \\
\frac{\partial \text{Cost}}{\partial b_2} &= (\text{sigmoid}(z) - y) \cdot x_2
\end{aligned}
$$

Substituting the gradient expressions

$$
\begin{aligned}
b_0 &= b_0 - \alpha \cdot (\text{sigmoid}(z) - y) \\
b_1 &= b_1 - \alpha \cdot (\text{sigmoid}(z) - y) \cdot x_1 \\
b_2 &= b_2 - \alpha \cdot (\text{sigmoid}(z) - y) \cdot x_2
\end{aligned}
$$

Questions

1. Apply simple linear regression with gradient descent method for x= {1, 2, 4, 3, 5} and y= {1, 3, 3, 2, 5} and plot Error (y axis) vs. Iteration (x axis) for 4 epochs (20 iterations). Manually work out the values for the first epoch and verify the results in your notebook.

2. Apply logistic regression with gradient descent method for following data set and plot Error (y axis) vs. Iteration (x axis) for 4 epochs (32 iterations). Manually work out the values for first epoch and verify the results in your notebook.

Hours of Study (X)	Pass (Y)
1	0
2	0
3	0
4	0
5	1
6	1
7	1
8	1

Here, X is the number of hours of study, and Y is the outcome (0 for fail, 1 for pass).

3.

	salary	experience
0	1.7	1.2
1	2.4	1.5
2	2.3	1.9
3	3.1	2.2
4	3.7	2.4
5	4.2	2.5
6	4.4	2.8
7	6.1	3.1
8	5.4	3.3
9	5.7	3.7
10	6.4	4.2
11	6.2	4.4

Create the following data set for Experience and Salary in CSV. Applying SLR, explore the relationship between salary and experience with exerience in x-axis and salary in y axis.
a. Check for various values of beta (slope) = 0.1, 1.5, and 0.8 with a fixed value of intercept i.e b=1.1. Plot the graph between beta and mean squared error(MSE) for each case.
b. Try with beta between 0 to 1.5 with an increment of 0.01 keeping b (intercept) as constant and Plot the graph between beta and mean squared error(MSE).
c. Try with different values of intercept for slope beta between 0 to 1.5 with an increment of 0.01. Plot the graph between beta and mean squared error(MSE).
d. Use the scikit learn and compare the results of MSE.

4. Apply Stochastic Gradient Descent for the afore-mentioned dataset, and arrive at different values of B0, B1 and error for 60 iterations of 5 epochs.

a. Plot the graph of log loss/error versus iteration.
b.Use the scikit learn and arrive at the results of B0, B1 and error, for 60 iterations of 5 epochs.
c. Plot the graph between beta (X-axis) and log loss/ error (Y-axis) using scikit learn and your approach separately.
d. Plot the separate graph of –log(x) (y=1 case) and –log(1-x) (y=0 case) and also draw the combined graph of .both cases.

5. Consider positive and negative slope dataset given below. Apply simple linear regression with gradient descent and illustrate the difference between slope values for both cases at different iterations. Plot the graph of slope(x-axis) vs MSE (y-axis) for both case separately.

```
x = np.array([1, 2, 4, 3, 5])
y = np.array([1, 3, 3, 2, 5])  # Positive slope

x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 8, 6, 4, 2])  # Negative slope
```

Additional Questions

1. Apply scikit learn model for Simple Linear regression using SGD of the given Salary_Data.csv dataset, and arrive at different values of B0, B1 and error for varying iterations. Plot the graph of epoch(X-axis) versus error(Y-axis).

2. Consider positive and negative slope dataset given below. Apply logistic regression with gradient descent and illustrate the difference between slope values for both cases at different iterations. Plot the graph of slope(x-axis) vs log-loss (y-axis) for both case separately.

```
x = np.array([1, 2, 3, 4, 5])
y = np.array([0, 0, 1, 1, 1])  # Positive slope

x = np.array([1, 2, 3, 4, 5])
y = np.array([1, 1, 0, 0, 0])  # Negative slope
```