UNIT – 1

TRAINING THE NETWORK-LOSS FUNCTIONS

3 KEY LOSS FUNCTIONS

1. Mean Squared Error Loss Function
2. Cross-Entropy Loss Function
3. Mean Absolute Percentage Error

1. Mean Squared Error Loss Function

Mean squared error (MSE) loss function is the sum of squared differences between the entries in the prediction vector y and the ground truth vector y_hat.

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

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

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

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.

An example of a regression problem would be predictions of . . .

- the number of products needed in a supply chain.
- future real estate prices under certain market conditions.
- a stock value.

2. Cross-Entropy Loss Function

Regression is only one of two areas where feedforward networks enjoy great popularity. The other area is classification.

In classification tasks, we deal with predictions of probabilities, which means the output of a neural network must be in a range between zero and one. A loss function that can measure the error between a predicted probability and the label which represents the actual class is called the cross-entropy loss function.

One important thing we need to discuss before continuing with the cross-entropy is what exactly the ground truth vector looks like in the case of a classification problem.

$$\hat{\vec{y}} = \begin{pmatrix} 0 \\ . \\ . \\ . \\ 1 \\ . \\ . \\ . \\ 0 \end{pmatrix} \qquad \vec{y} = \begin{pmatrix} 0.52 \\ . \\ . \\ . \\ 0.12 \\ . \\ . \\ . \\ 0.78 \end{pmatrix}$$

One-hot-encoded vector (left) and prediction vector (right).

The label vector y_hat is one hot encoded which means the values in this vector can only take discrete values of either zero or one. The entries in this vector represent different classes. The values of these entries are zero, except for a single entry which is one. This entry tells us the class into which we want to classify the input feature vector x.

The prediction y, however, can take continuous values between zero and one.

Given the prediction vector y and the ground truth vector y_hat you can compute the cross-entropy loss between those two vectors as follows:

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

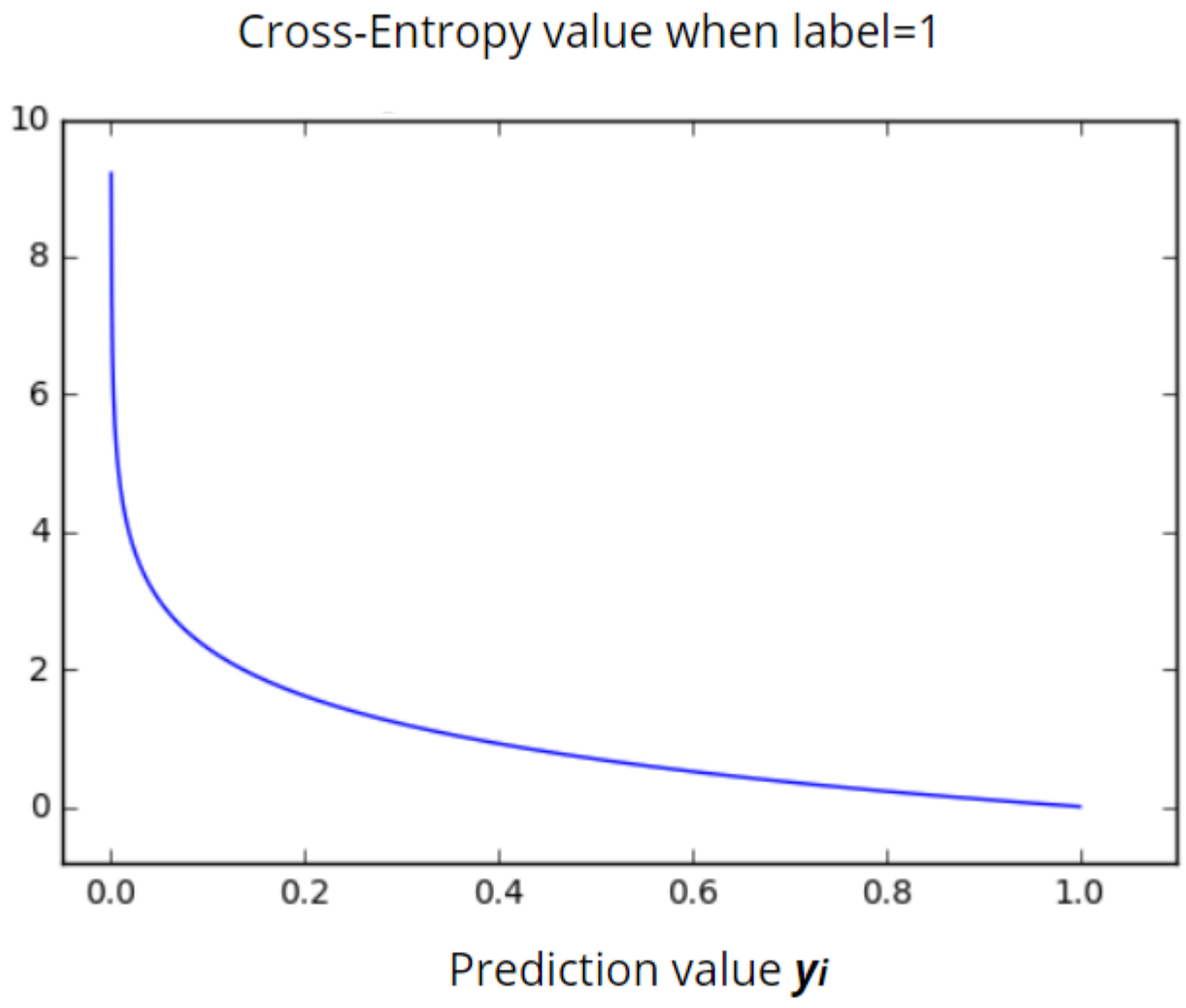
y_i : entries in the prediction v

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

Cross-entropy loss function

First, we need to sum up the products between the entries of the label vector y_hat and the logarithms of the entries of the predictions vector y. Then we must negate the sum to get a positive value of the loss function.

One interesting thing to consider is the plot of the cross-entropy loss function. In the following graph, you can see the value of the loss function (y-axis) vs. the predicted probability y_i. Here y_i takes values between zero and one.



Cross-entropy function depending on prediction value.

We can see clearly that the cross-entropy loss function grows exponentially for lower values of the predicted probability y_i. For y_i=0 the function

becomes infinite, while for y_i=1 the neural network makes an accurate probability prediction and the loss value goes to zero.

3. Mean Absolute Percentage Error

Finally, we come to the Mean Absolute Percentage Error (MAPE) loss function. This loss function doesn't get much attention in deep learning. For the most part, we use it to measure the performance of a neural network during demand forecasting tasks.

First thing first: what is demand forecasting?

Demand forecasting is the area of predictive analytics dedicated to predicting the expected demand for a good or service in the near future. For example:

- In retail, we can use demand forecasting models to determine the amount of a particular product that should be available and at what price.
- In industrial manufacturing, we can predict how much of each product should be produced, the amount of stock that should be available at various points in time, and when maintenance should be performed.
- In the travel and tourism industry, we can use demand forecasting models to assess optimal price points for flights and hotels, in light of available capacity, what price should be assigned (for hotels, flights), which destinations should be spotlighted, or, what types of packages should be advertised.

Although demand forecasting is also a regression task and the minimization of the MSE loss function is an adequate training goal, this type of loss

function to measure the performance of the model during training isn't suitable for demand forecasting.

Why is that?

Well, imagine the MSE loss function gives you a value of 100. Can you tell if this is generally a good result? No, because it depends on the situation. If the prediction y of the model is 1000 and the actual ground truth label y_hat is 1010, then the MSE loss of 100 would be in fact a very small error and the performance of the model would be quite good.

However in the case where the prediction would be five and the label is 15, you would have the same loss value of 100 but the relative deviation to the ground-truth value would be much higher than in the previous case.

This example shows the shortcoming of the mean squared error function as the loss function for the demand forecasting models. For this reason, I strongly recommend using mean absolute percentage error (MAPE).

The mean absolute percentage error, also known as mean absolute percentage deviation (MAPD) usually expresses accuracy as a percentage. We define it with the following equation:

$$MAPE = \frac{100\%}{N} \sum_{i=0}^{N} \frac{|y_i - \hat{y}_i|}{\hat{y}_i}$$

In this equation, y_i is the predicted value and y_hat is the label. We divide the difference between y_i and y_hat by the actual value y_hat again. Finally, multiplying by 100 percent gives us the percentage error.

Applying this equation to the example above gives you a more meaningful understanding of the model's performance. In the first case, the deviation from the ground truth label would be only one percent, while in the second case the deviation would be 66 percent:

$$MAPE_1 = 100\% \left| \frac{1000 - 1010}{1000} \right| = 1\,\%$$

$$MAPE_2 = 100\% \left| \frac{5 - 15}{15} \right| = 66\,\%$$

We see that the performance of these two models is very different. Meanwhile, the MSE loss function would indicate that the performance of both models is the same.

Reference:

https://builtin.com/machine-learning/loss-functions