Understanding different Loss Functions for Neural Networks

Shiva Verma
Jun 20 · 4 min read ★

Image by KotaFeja

There are different loss functions available for different objectives. In this mini blog, I will take you through some of the very frequently used loss functions, with a set of examples. This blog is designed by keeping the Keras and Tensorflow framework in the mind.

Loss Function | Brief Intro

Loss function helps in optimizing the parameters of the neural networks. Our objective is to minimize the loss for a neural network by optimizing its parameters(weights). The loss is calculated using loss function by matching the target(actual) value and predicted value by a neural network. Then we use the gradient descent method to optimize the weights of the network such that the loss is minimized. This is how we train a neural network.

• • •

Mean Squared Error

When you have a regression task, one of the loss function you can go ahead is this one. As the name suggests, this loss is calculated by taking the mean of squared differences between actual(target) and predicted values.

For Example, you have a neural network with which takes some data related to house and predicts it's price. In this case, you can use the `MSE` loss. Basically, in the case where the output is a real number, you should use this loss function.

Mean Squared Error

Binary Crossentropy

When you have a binary classification task, one of the loss function you can go ahead is this one. If you are using `BCE` loss function, you just need one output node to classify the data into two classes. The output value should be passed through a *sigmoid* activation function so the output is in the range of (0–1).

For example, you have a neural network which takes data related to atmosphere and predicts whether it will rain or not. If the output is greater than 0.5, the network classifies it as `rain` and if the output is less than 0.5, the network classifies it as `not rain`. (it could be opposite depending upon how you train the network). More the probability score value, more the chance of raining.

Binary Crossentropy

While training the network, the target value fed to the network should be 1 if it is *raining* otherwise 0.

One important thing, if you are using `BCE` loss function the output of the node should be between (0–1). It means you have to use sigmoid activation function on your final output. Since sigmoid converts any real value in the range between (0–1).

What if you are not using sigmoid activation on the final layer? Then you can pass an argument called `from logits` as `true` to the loss function and it will internally apply the sigmoid to the output value.

Categorical Crossentropy

When you have a multi-class classification task, one of the loss function you can go ahead is this one. If you are using `CCE` loss function, there must be the same number of output nodes as the classes. And the final layer output should be passed through a *softmax* activation so that each node output a probability value between (0–1).

For example, you have a neural network which takes an image and classifies it into a cat or dog. If cat node has high probability score then the image is classified into cat otherwise dog. Basically, whichever class node has the highest probability score, the image is classified into that class.

Categorical Crossentropy

For feeding the target value at the time of training, you have to one hot encode them. If the image is of cat then target vector would be (1, 0) and if the image is of dog, target vector would be (0, 1). Basically, target vector would be of the same size as the number of classes and the index position corresponding to the actual class would be 1 and all other would be zero.

What if you are not using *softmax* activation on the final layer? Then you can pass an argument called `from logits` as `true` to the loss function and it will internally apply the *softmax* to the output value. Same as in the above case.

Sparse Categorical Crossentropy

This loss function is almost similar to `CCE` except for one change.

When you are using `SCCE` loss function, you do not need to one hot encode the target vector. If the target image is of a cat, you simply pass 0, otherwise 1. Basically, whichever the class is you just pass the index of that class.

Sparse Categorical Crossentropy

• • •

These were the most important loss functions. And probably you will be using one of these loss functions when training your neural network.

Following are the source code for all available loss function in Keras.

keras-team/keras
Deep Learning for humans. Contribute to keras-team/keras development by…
github.com

Machine Learning Artificial Intelligence Deep Learning Neural Networks
Loss Function