Assignment 1 Report

Advanced Concepts in Machine Learning

Implementing Backpropagation

Foti Kerkeshi

November 2022

Implementation

For the neural network implementation, I have based all the operations on NumPy arrays. After I created the dataset that contains the same data as inputs and output, I created a Layer class. This class holds all the weights and biases of a NN layer.

After this, I implemented the Mean Squared Error loss function and its derivative function. The next step was to implement the forward and backward propagation. The first is very straightforward, as the only operation applied is dot product between matrices and after the first layer, I applied ReLu non-linearity.

For the backpropagation algorithm, I manually computed the gradients for each layer (including here the activation layer - ReLU). The main *forward_backward* function returns all the data needed to do the parameter update for each step including here the gradients of the parameters, the calculated loss function, and the predicted output. As an optimizer SGD was used with different learning rates, and their results are explained in more detail in the experiments section.

Experiments

Initially, I was training the neural network with a few steps ranging from 1000 to 5000, but it was not enough as the training was being stopped prematurely. In addition, the predicted output was far from the ground truth, and the model was increasing the target value of 2-3 scalars out of 8, which was the size of the output array.

Next, I increased the number of steps to 20,000 - 50,000 I started getting better results, however, the loss function plot was not very smooth. The learning rate since the beginning of the experiments was 0.001. After getting not very satisfactory results I decided to decrease the learning rate to 0.00001 and this is when the results were comparingly better than earlier. The loss function plot is shown below.

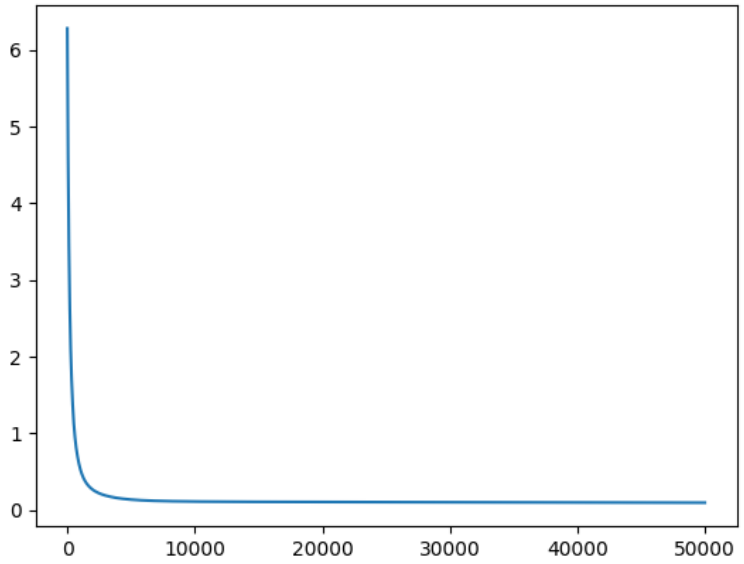

Figure 1 *- Loss function, learning rate - 0.01, steps - 50,000*

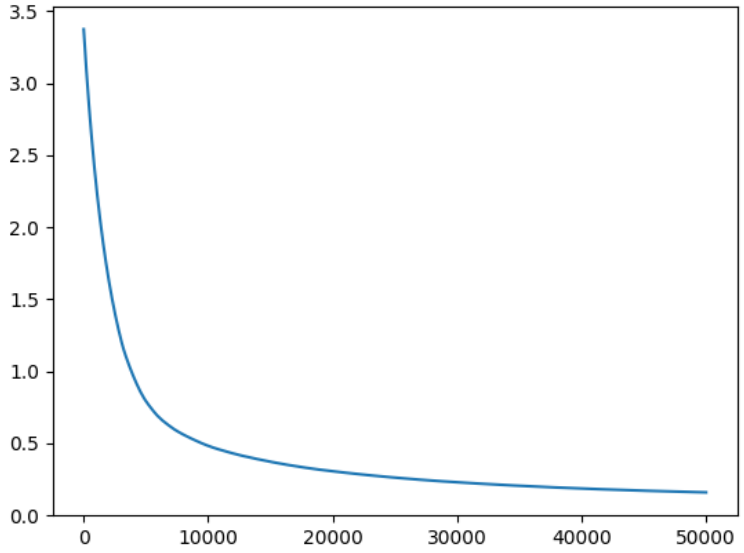

Figure 2 *- Loss function, learning rate - 0.00001, steps - 50,000*

The results

Analyzing the previous results I came to the conclusion that a very small learning rate is required for the neural network to perform well. The value 0.0001 was the best learning parameter for the optimizer and the model was converging to after around

Regarding the learned weights, as expected the predicted vectors that contain high values are associated with high gradient values in all the neural network parameters. Another experiment was to train the model without using bias parameters which inferred very high values in the output vector. Observing the bias parameters from the previous experiments most of them, in particular, the bias parameters of the first layer were always negative and they helped regularize the prediction.

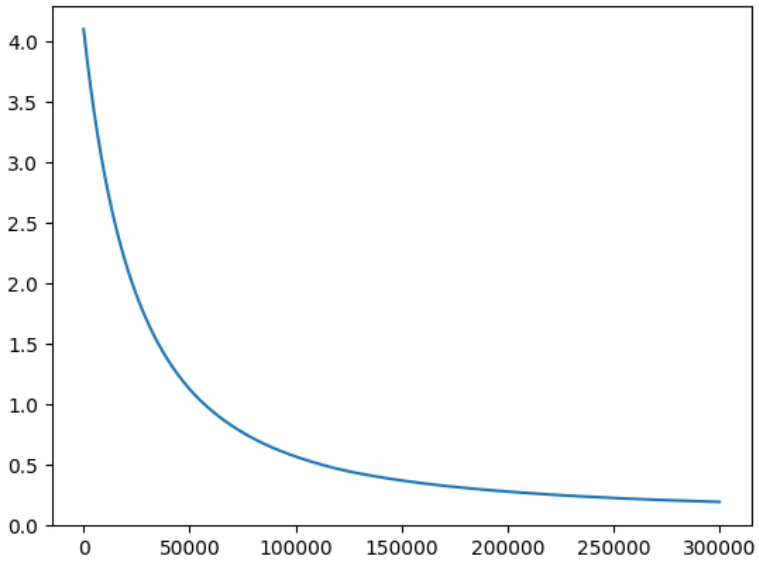

Figure 3 *- [Extra Trial] - low learning rate, slows down the convergence of the loss function. Here, learning rate - 0.000001, steps - 300,000*