ML_HW3_Report

Name:張浚騰

Deadline: 2021 年 12 月 2 日(四) 23:59

I. Describe what problems you encountered when implementing the basic functions.

Deep learning is more like doing matrices multiplication. I think most people might forget about the dimension issue, which may cause bugs. You must transpose the matrices to the right way so that you can implement your neural network properly. And every layer has different weight matrices and bias, I think it is better to do math formula derivation on paper. Otherwise, you might got confused about the relationship between current layer and former layer.

II. Describe how you solve those problems.

Before implementing the neural network, I will do the formula derivation to make sure completely understand how it works. By doing so, it didn't really take much effort.

III. Briefly describe how you build the binary classifier

1. Define the layer of neural network. Parameter initialization using small random numbers is simple approach, but it guarantees good enough starting point for our algorithm.
2. Activation functions give the neural networks non-linearity. In our example, we will use sigmoid and ReLU. Sigmoid outputs a value between 0 and 1 which makes it a very good choice for binary classification. You can classify the output as 0 if it is less than 0.5 and classify it as 1 if the output is more than 0.5.
3. During forward propagation, in the forward function for a layer*l*you need to know what the activation function in a layer is (Sigmoid, tanh, ReLU, etc.). Given input signal from the previous layer, we compute Z and then apply selected activation function.
4. In order to monitor the learning process, we need to calculate the value of the cost function. We will use the below formula to calculate the cost
5. Backpropagation is used to calculate the gradient of the loss function with respect to the parameters. This algorithm is the recursive use of a "chain rule" known from differential calculus

6. The goal of the function is to update the parameters of the model using gradient optimization.
   By doing the above step, we could build the full model.

IV. Describe if you pay extra effort to improve your model (e.g. hyperparameter finetuning).

Since it is not a complicated neural network. It really does not take much effort to build. However, if you want to speed up the convergence rate increasing the learning rate might help but you need to be careful about stuck in local minimum.

Increasing the layer might Increasing the number of hidden layers much more than the sufficient number of layers will cause accuracy in the test set to decrease and cause your network to overfit to the training set, that is, it will learn the training data, but it won't be able to generalize to new unseen data.

V. Bonus part.

Just implement the bonus function and set training data and validation data to 8 : 2 and do the training.

Accuracy is highly dependent on the number of iteration, so I had tried many different iterations and learning rate to get the final result.