[COSE474] Deep Learning Project #1: MLP Implementation

2022320315 데이터과학과 김승주

 # Description of code

In the first part, I did forward pass of a neural network, which is the process of passing input data through the network to calculate class scores.

Then I compute the loss for a neural network with a Softmax classifier. 'p' represents the class probabilities, which are calculated using the Softmax function from the raw class scores (scores). The Softmax function normalizes the scores to obtain class probabilities.

And I perform the backward pass to compute the gradients for the weights and biases of the neural network. The code employs the chain rule of partial derivatives for this purpose. In this part, I tried to remember the dimension of the matrix.

And I use np.random.choice() function to create a random minibatch of training data and their corresponding labels. This random selection of examples introduces randomness into the optimization process and allows the neural network to generalize better.

Then I update the parameters using gradients calculated beforehand. It is important to add in a negative direction.

Finally I compute the forward pass of the neural network and returns the predicted class (index) for each input in 'X' based on the highest probability in the output scores.

 # Results

With the default parameters provided, I get a validation accuracy of about 0.27 on the validation set. And this isn't very good. First, I use hidden_sizes = [100, 300, 500], learning_rates = [1e-5, 1e-4, 1e-3], regs = [0.05, 0.15, 0.25], num_iters=1000, batch_size=200, learning_rate_decay=0.95. With grid search, I get a highest validation accuracy of **0.468 {'hidden_size': 300, 'learning_rate': 0.001, 'reg': 0.05} and test accuracy of 0.483**.

Then I try many other cases. And I finally get validation accuracy **0.502** {hidden_size: 800, num_iters=3000, batch_size=200, learning_rate=0.001, learning_rate_decay=0.95, reg=0.05}, test accuracy **0.506**.

 # Discussions

I improve my model's accuracy by changing some hyperparameters. Although it is quite high compared to the original case, it is still low. In normal cases, we can not only change hyperparameters, but also model selection. So I think we can improve accuracy by adding more layers, using other kind of model etc.