COSE474 Deep Learning
Project 1: MLP Implementation

컴퓨터학과 2022320006 최진혁

1. Description of code

In the class, we studied data flow of forward and backpropagation of softmax classifier. However, in the project 1, there are N taring samples and the shape of X is (N, D). Thus, the forward and backpropagation are performed on each training data X[i], and each result is stored into N row matrix. It can be seen like there are N forward & backward data flow, and all flow is averaged to 1 scalar loss value. The final loss is average N sample loss with L2 regularization loss.

A. TwoLayerNet.loss()

First, compute forward pass. Input of hidden layer is h = ReLU(z), where z is (N, H) matrix and z[i] is the output of FC layer with parameter = (W1, b1) on given input X[i]. Scores is (N, C) matrix, and is the output of FC layer with parameter = (W2, b2) on given input h. Scores[i][c] is the score for class c on input X[i].

Then, feed scores matrix into softmax and compute data loss. Entire output of softmax is stored into softmax_out for backpropagation, which has shape of (N, H). Each scores[i] is the probability for classes on input X[i], so scores[i] is forwarded to softmax respectively, and softmax_out[i][y[i]] is accumulated to data_loss. To avoid the overflow in the training phase, softmax equation has adjusted a little. For each scores[i], the max value is subtracted from all scores[i][j]. Then, data_loss is divided by sample size N, and the L2 regularization loss reg_loss is computed. The final loss is sum of data loss and regularization loss.

In backpropagation phase some trick is used. First, one-hot lable y_one_hot is computed to use gradient rule of softmax classifier discussed in the class. Then on the data loss L, dL/dscores = dL/dL_i * (softmax_out – y_one_hot) where dL/dL_i = 1/N. Then backpropagation flow that is discussed in the class can be performed on each N training sample X[i]. It is important to accumulate gradients for parameters because each parameter has passed copy gate when they applied to each N training sample. Thus, the gradient for W and b having shape of (N, …) is accumulated on axis 0 and then transposed if needed. Especially, gradient of weights on the regularization loss also be accumulated to final gradient for weights. The gradient is reg * 2 * W.

B. TwoLayerNet.train()

First, get the minibatch using np.random.choice. For matching input and label data, first get the random indices and then get minibatch.

The parameter update can be performed using gradient descent subtracting (learning rate * grads[parameter_name]) from each parameter.

C. TwoLayerNet.predict()

Prediction is simply performing forward pass and returning the maximum score's index of output. This can be performed using np.argmax() function.

2. Result & Discussion

The first validation accuracy of the model is about 0.27, and 0.241 in my case. This is poor performance. For hyperparameter tuning, the grid search is performed to find optimal value. The target hyperparameters are hidden layer size (128, 256), learning rate (0.0001, 0.0005, 0.001), and regularization strength (0.2, 0.3, 0.4). It is found that the model with 256 hidden layers, 0.01 learning rate, and 0.4 regularization strength is best among these. The visualization of parameters get more complicated rather than the first model.

However, the best test accuracy is 0.469, which is still low. This is because of the model complexity. Although we can grow the hidden layer size, the number of hidden layer is always 1. Two layer MLP is too simple to handle CIFAR-10 dataset. More deep MLP model can increase the accuracy. On the other hand, the training phase in this project took some time. If we try to train more deep model, we must use GPU to train.