Machine Learning

Assignment: Neural Network

Due: Monday, July 11th at 11:59pm

I. Purpose:

The goal of this assignment is to get familiarised with building simple neural network models with fully-connected layers to perform classification and test it out on the CIFAR-10 dataset. **Please finish this homework using Google Colab**. Feel free to use Pycharm for task 1 and 2 to edit, test, and debug your code.

- Learn how to design your neural network.
- Learn how to perform forward pass and backward pass for your training.
- Learn how to train your model for classification.
- Learn how to tune your hyperparameters and other technologies to improve the performance.

II. Link of Data and Code:

https://drive.google.com/drive/folders/1X3lGDvx189n0GWfmnF4KjLklWObQY0xi?usp=sharing

III. Description

Task 1: Design Your Neural Network

In this task, you will create a two-layer fully-connected neural network for the classification task. This network has an input dimension of N, a hidden layer dimension of H, and need to achieve classification output over C classes. You will complete the forward pass, backward pass function for updating the parameters in your model. All your functions will be completed in the corresponding position in network.py. You may go through the HW.ipynb Task. 1 and finish the implementation.

- Finish the forward pass function. (detailed grading terms please see Grading and Submission)
  - Create the forward pass, calculate the class scores.
  - Complete the forward pass, calculate the classifier loss.
- Finish the backward pass function (detailed grading terms please see Grading and Submission)
  - Complete the backward pass, calculate the derivatives of the weights.
- Go through the ipynb and show the result in each part. (detailed grading terms please see Grading and Submission)

Task 2: Network Training

Once you complete your network design, you may train your network. You will use both a toy dataset and CIFAR-10 dataset (http://www.cs.toronto.edu/~kriz/cifar.html) to train your model. All your functions related to the training procedure will be completed in the corresponding position in network.py. You may go through the HW.ipynb and finish the implementation.

- Finish the train function. (detailed grading terms please see Grading and Submission)
  - Create a minibatch of the training dataset randomly.
  - Create gradient descent to update the parameters of the network.

- Finish the prediction function. (detailed grading terms please see Grading and Submission)
    - Complete the prediction function.
- Go through the ipynb and show the result in each part. (detailed grading terms please see Grading and Submission)

Task 3: Tune Your Network

After you get your prediction model, you may find that the loss is decreasing more or less linearly, which seems to suggest that the learning rate may be too low. Moreover, there is no gap between the training and validation accuracy, suggesting that the model we used has low capacity, and that we should increase its size. In this task, you may tune your network to achieve a better classification accuracy on the validation set. All the code should be completed in the corresponding position in the HW.ipynb

- Implement your own techniques to improve the performance of your model (detailed grading terms please see Grading and Submission)
    - Tune hyperparameters or use any techniques to improve the performance using the training set and validation set.
    - show your result on the testing set.

IV. Grading and Submission

- The assignment will be evaluated in a total of 70 points. The basic scores are generally given based on the following table.

Part 1: Design Your Neural Network (25 points)

- Create the forward pass, calculate the class scores. (5')
- Complete the forward pass, calculate the classifier loss (5')
- Complete the backward pass, calculate the derivatives of the weights (5')
- Compute and show "Difference between your scores and correct scores" (5')
- Compute and show "max relative error" (5')

Part 2: Network Training (25 points)

- Create a minibatch of training dataset randomly (5')
- Use minibatch GD to update the parameters of the network (5')
- Complete the prediction function (5')
- Use toy data and plot the correct loss history (5')
- Load CIFAR-10 to train the model, display the validation accuracy (5')

Part 3: Tune Your Network (20 points)

- Use one technology (e.g., tune learning rate, increase model capacity etc.) to improve your performance (5')
- Explain the technologies you used with >=50 words (e.g., which parameters are changed, what are the new strategies, and reasoning) (5')
- Plot the loss function and train/val accuracies, visualize the weights of the best network (5')
- Display your final result on the test set, show the classification accuracy (5')

For each 5' scale.
5' = perfectly correct

4’ = minor flaw
3’ = mostly incorrect
2’ = totally incorrect
1’ = do something
0’= not do anything

- The assignment should be submitted with two files:
  i) A single PDF report file should be submitted to Brightspace with 1) week number, 2) last name, and 3) VUID (e.g., “Week01_Huo_huoy1.pdf”). The ideal PDF report file is a printed Colab ipynb file with required results embedded.
  ii) All source code should be submitted to Brightspace as a single zip file with last name and VUID (e.g., “Week01_Huo_huoy1.zip”).