CSCE 5218 & 4930: Deep Learning: Homework 1

Due: 03/25/2022, 11:59pm
Point: 100

You will do this assignment using Google's Colab service (check how to use it at here). Your code will be included in a Jupyter notebook and will be run on the cloud, where you have access to GPUs for free with your Google account. To make a new notebook, go to File → New notebook in the above link. Then add both code and text snippets; use text snippets to explain what your code does, and state which part this snippet is implementing. In Part B, also use text snippets to explain your observations about the results you obtain.

Turn on the GPU mode in Edit → Notebook settings and set Hardware accelerator to GPU.

1. Building a Custom Data Loader (50 points)

In our PyTorch tutorial, we show how to use the `torchvision` library to load common dataset such as MNIST. However, in real world applications, it's more common for you to deal with custom data. In this exercise, you are provided with a bunch of images and you need to write a data loader to feed these images to your model in PyTorch.

Note: You are not allowed to use `torchvision` in this exercise. You can refer to the example in our lecture. You may check another reference about data loaders by clicking here.

1.1. Instruction

- Please download the dataset from this link. There are two directories within the given zip file, one for training samples and one for test. In each directory, there are 10 subdirectories each containing the images that belong to that category. The structure looks like:

  ```
  cifar10_train/airplane/airplane_00001.png
  cifar10_train/airplane/airplane_00002.png
  ......
  cifar10_train/truck/truck_04999.png

  cifar10_test/airplane/airplane_00001.png
  cifar10_test/airplane/airplane_00002.png
  ......
  cifar10_test/truck/truck_00999.png
  ```

- Implement a class `CifarDataset`, which represents the given dataset. Your custom dataset should inherit `torch.utils.data.Dataset`. This class can be used for both training set and test set, which depends on the argument root_dir passed to the constructor.

- Once you set up your custom dataset, you can feed it to a `torch.utils.data.DataLoader` so it helps to iterate through your dataset.

Tip: Check the example in our PyTorch tutorial.

2. Training a neural network in PyTorch (50 points)

In this part you need to implement a 3-layer MLP model (one input layer, one hidden layer with tanh activation and one output layer) in PyTorch (named `MultiLayerPerceptronModel`, which will be used to classify the images from the dataset in Part A. You can use the built-in modules in PyTorch to build your model, such as Linear, Dropout, Tanh, etc. You also need to write the training function (`training`), and should explore the following hyperparameter settings:

- Batch size: Number of examples per training iteration.
- Hidden size: Try using different number of hidden nodes in your model and compare the performances.
- Dropout: Dropout is an effective strategy to defend against overfitting. Add a dropout layer after the hidden layer, and try using different dropout rate to compare the performances. Check Dropout in PyTorch at here.
- Optimizer: Try using different optimizers such as `SGD` and `Adam`.
- Learning rate: Learning rate is key hyperparameter in model training, and you can gradually decreasing the learning rate to further improve your model. Try using different learning rate to compare the performance.

To get full credit, you should explore at least 3 different types of hyperparameters (from the 5 listed above), and choose at least 3 different values for each hyperparameters. For simplicity, you could analyze one hyperparameter at a time (i.e. fixing all others to some reasonable value), rather than perfoming grid search.

To evaluate the performance of trained models, you also need to write a function (`evaluation`) which loads the trained model and evaluates its performance on train/test set.

Please include a brief report (a PDF file) with your Jupyter notebook, in which you clearly state what hyperparameters you explored, and what accuracy the model achieved on the train/test set for each setting of these hyperparameters.

3. Submission

Your submission should contain two files, a Jupyter notebook file and a PDF file.

In the Jupyter notebook, with parts clearly labeled, making sure to include the following classes/functions:

- `class CifarDataset`
- `class MultiLayerPerceptronModel`
- `function training`
- `function evaluation`

In the PDF file, you should discuss the performance of your network. Also, compare performance using different hyperparameters.

Acknowledgement. The assignment is adapted from M. Zhang at University of Pittsburgh.