Homework-6: PyTorch

Deadline: November 7th, 11:59PM ET.

In this homework, you are expected to practice PyTorch, including building datasets and models and writing training loops with best practices. You may execute this assignment on your local machine, on the cloud, or in online python environments (e.g. CoLab).

Your homework submission should be on GitHub. Use the following GitHub classroom to access the assignment and create your assignment repository:
https://classroom.github.com/a/wHbE2LGk

You should submit the URL for your GitHub repository on Canvas. Grading penalty will be applied if otherwise.

Cite any external sources you use (the lecture materials don't count as external sources). External sources shouldn't exceed more than 30% of the final solution.

Submit your answers in one (or more) Jupyter notebook(s) **but clearly identify in your ReadMe file the location of the answers for every question**. If you need to include output or screenshots, you may include them in your notebook or attach them as separate image files using the naming convention qx.png (where x is the question number). If you have more than one image for a given question, name the files as qx_1.png, qx_2.png, etc.

In this homework, you will use PyTorch for the NSL-KDD dataset to build neural networks and write training loops to predict not just whether there is an attack, but also the type of the attack. In other words, this is a multi-class classification problem with 5 possible categories (normal, DOS, R2L, U2R, probing)

Using Spark

Q1 (10%) Prepare the data and convert them to 3 Datasets in PyTorch for training, validation, and testing respectively. You may use KDDTrain+.txt as the training dataset, 50% of KDDTest+.txt as the validation dataset, and the remaining 50% as the test dataset. If during the data preparation, an error occurs due to the size of the dataset, you may use a subset of these datasets.

Q2 (10%) Build a Neural Network via subclassing nn.Module. You may choose your own activation function, number of layers, and number of neurons in each layer. Include a screenshot of the printout of your model (that is, print(mymodel) where mymodel is your model instance).

Q3 (30%) Write a training loop that contains at least the following features: conduct validation for each epoch; compute the train/validate loss/metric for each epoch; print out the progress of each epoch; save the current best model so far at every epoch. You may choose your own metric.

Further, after the training loop, plot three figures

- The train loss per SGD iteration, that is, the train loss of every batch across all batches in all epochs. The horizontal axis should range from 1 to the total number of train iterations, which equals (number of epochs)*(number of batches in training data).
- The train and validate loss across different epochs.
- The train and validate metric across different epochs.

Please select your own hyper-parameters and conduct a run of the training loop, and in your submission, include a screenshot of the printout of the training loop and the three figures.

Q4 (30%) Please tune the training hyper-parameters (learning rate/batch size/number of epochs/choice of optimizer/others) such that your model converges well. Please try at least 3 sets of hyper-parameters, and document the results (the values of the hyper-parameters, screenshots of the printout of the training loop, and the three figures as in Q3). Please provide a short description on your tuning process, including the rationale for choosing the hyper-parameters.

Q5 (20%) Load back the best model encountered in your best run in Q4, and calculate the metric on the test data set (please include a screenshot). Note that when evaluating the metric, the gradient is not needed, and thus, building the computation graph is not necessary. What should you do to prevent PyTorch from building the computation graph in the evaluation process? Please explain your answer and include a screenshot of the relevant code.