LING572 HW9: Neural Networks
Due: 11pm on March 19, 2020

This assignment explores text classification using a version of the Deep Averaging Network from Iyyer et al 2015, as discussed in the first half of the Recurrent Neural Network lectures on applying MLPs to text classification. Through the assignment, you will:

- Get familiar with the basics of doing text classification in PyTorch. We will "pseudo-reproduce" a result from Iyyer et al by training a Deep Averaging Network on the IMDb Reviews dataset for sentiment classification. Its a pseudo-reproduction for a few reasons:
  - We are using a slightly different dataset split (17.5k train instead of 25k)
  - Our model will have 2 hidden layers instead of 3, just for compute efficiency
- Implement a linear layer, the basic building block of neural networks.
- Implement L2 regularization, and see how it impacts performance and runtime.
- Implement early stopping.

We have provided the bulk of the necessary code. You will have to fill in some blank spots for each of the implementation questions, described in more detail below.

In the directory `/dropbox/19-20/572/hw9/` you will find the following:

- env: the environment; no need to touch this
- data: the IMDB dataset
- model.py: defines a linear layer and a Deep Averaging Network
- main.py: the main script for building and training a model

  NB: at the bottom of the file, you will see many command-line arguments. `python main.py` will run with the defaults, but you can also set these. For example, to train for 10 epochs, you can use `python main.py --num_epochs 10`.
- run_hw9.sh: an example executable to modify / submit via condor. Note: you will want to change the '`cd ...`' line to point to your directory. Note that this file shows you how to activate the supplied conda environment.

Q0 (10 points): Install Anaconda. This is a necessary component for running the code for this assignment. I have setup a conda environment for the assignment, but you will need to install anaconda in order to use that environment. These are "free points". From your home directory, please execute the following steps:

1. `wget https://repo.anaconda.com/archive/Anaconda3-2019.10-Linux-x86_64.sh`
2. `sh Anaconda3-2019.10-Linux-x86_64.sh`

The first two lines of `run_hw9.sh` show how to now activate the environment that we have supplied with all of the necessary libraries.

Q1 (20 points): Implement the forward pass of a LinearLayer and train a model.

- Find the '# TODO:' comment in `/dropbox/19-20/572/hw9/model.py`. Implement the .forward method there, following the instructions in the comment and the docstring.
- Run `python main.py --num_epochs 6 > q1.out`. Please fill out the information below:

Epoch num	Train loss	Dev loss
0		
1		
2		
3		
4		
5		

Test set accuracy of best model:

Total runtime:

Q2 (20 points): L_2 regularization prevents over-fitting by penalizing large weight values. In particular, if $\mathcal{L}(\theta)$ is our loss function, L_2 regularization replaces that loss with

$$\mathcal{L}'(\theta) = \mathcal{L}(\theta) + \lambda \|\theta\|_2^2$$

where $\|\theta\|_2^2$ is the squared L^2 norm (i.e. the sum of the squares of all parameters in θ). You will:

- Add L_2 regularization in `main.py`. Search for '# TODO:' and find the one right above the line `L2 = 0.0`. Replace this with the squared L_2 norm of the model's parameters.

  Hint: `model.parameters()` returns an iterator over the parameters, which are each a `torch.tensor`.

  Note: you should use the `--L2` flag (stored in `args.L2`) to only compute the regularization term when requested from the command-line.

- Run `python main.py --num_epochs 6 --L2 > q2.out`, training for 6 epochs with L_2 regularization. Please fill out the information below:

Epoch num	Train loss	Dev loss
0		
1		
2		
3		
4		
5		

Test set accuracy of best model:

Total runtime:

Q3 (20 points): Another method for preventing over-fitting is early stopping. On this approach, we define a hyper-parameter patience (p), which is an integer. We then train for a large number of epochs, but if loss on the dev set is worse at epoch t than at epoch $t-p$ for any epoch, we stop training immediately.

- Implement early stopping. The final '# TODO:' in `main.py` occurs instructs you to implement this early stopping protocol in the main training loop.

  Note: you may need to edit code outside the immediate 'if' statement that the comment appears in.

- Run `python main.py --num_epochs 12 --patience 3 --L2 > q3.out` and fill out the information below (if early stopping stops your model before 12 epochs, you will have empty rows in this table, which is acceptable):

Epoch num	Train loss	Dev loss
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		

  Test set accuracy of best model:

  Total runtime:

Q4 (5 points): We will issue 5 total points based on the runtimes reported in Q1-3. They should be approximately no more than 10 minutes when running on patas. No need to submit anything for this question.

Submission: Submit the following to Canvas:

- Your note file *readme.(txt | pdf)* that includes the tables and additional information above, and any notes that you want the TA to read.
- hw.tar.gz that includes all the files specified in dropbox/19-20/572/hw9/submit-file-list, plus any source code (and binary code) used by the shell scripts.
- Make sure that you run **check_hw9.sh** before submitting your hw.tar.gz.