CSE 597 - Assignment 1
Minimalist Neural Networks

Overview

This is an exercise in developing a minimalist neural network toolkit for NLP. Your implementation will be used to construct a Deep Averaging Network for text classification on Stanford Sentiment Treebank.

Submission Instruction

Follow these steps exactly, so the Gradescope autograder can grade your homework. Failure to do so will result in a zero grade:

1. You *must* download the homework template files from Canvas. Each template file is a python file that gives you a headstart in creating your homework python script with the correct function names for autograding.
2. Upload your **minnn.py** file to Gradescope by its due date.

Environment Setup

This assignment requires **Python 3.8**. The only allowed external library is **numpy==1.19**, no other external libraries are allowed.

File Structure

The starter code contains the following

1. **minnn.py**: **This is what you'll need to implement.** It implements a very minimalist version of a dynamic neural network toolkit (like PyTorch or Dynet). Some code is provided, but important functionality is not included. Please read the documentation in minnn.py and the instruction below carefully to understand what needs to be done.
2. **test_minnn.py**: This script can test your minnn.py implementation.
3. **classifier.py**: training code for a Deep Averaging Network for text classification using minnn. You can feel free to make any modifications to make it a better model. *However,* the original version of `classifier.py` will also be tested and results will be used in grading, so the original `classifier.py` must also run with

your `minnn.py` implementation. Using the option of `--do_gradient_check 1` when running `classifier.py` to do gradient checking overall. (Remember to turn this option off in your submitted `classifier.py`.)

4. **data**/: Stanford Sentiment Treebank (SST) with tree info removed.

Your Tasks

You will need to finish the implementation of **minnn.py**. Some code is provided, but important functionality is not included. Please read the documentation in this file carefully to understand what needs to be done. You need to implement the following functions and classes:

- `def accumulate_grad(self, g: np.ndarray)`
- `def accumulate_grad_sparse(self, gs: List[Tuple[int, np.ndarray]])`
- `class OpLookup(Op)`
- `class OpDot(Op)`
- `class OpTanh(Op)`
- `class OpMax(Op)`
- `def xavier_uniform(shape: Sequence[int], gain=1.0)`
- `class MomentumTrainer(Trainer)`

Command Line

After you finish your implementation of minnn.py, train your neural networks on SST dataset by running

```
$ python classifier.py --train=data/sst-train.txt
--dev=data/sst-dev.txt --test=data/sst-test.txt
--dev_out=outputs/sst-dev-output.txt
--test_out=outputs/sst-test-output.txt
```

Reference accuracies: with our implementation and the default hyper-parameters, the mean(std) of accuracies with 10 different random seeds on sst is dev=0.4031(0.0105), test=0.4070(0.0114). If you implement things exactly in our way and use the default random seed and use the same environment (python 3.8 + numpy 1.19), you may get the accuracies of dev=0.4015, test=0.4109.