CS224d: Deep Learning for Natural Language Processing

Assignment #2: Deep and Recurrent Neural Networks

Due Date: 4/30 (Thu) 11:59 PM PST.

New! Submission instructions posted below.

Update (4/29): Starter code is updated, fixing a bug that affected certain sparse gradient updates. This bug could cause the gradient check to fail for L, even if your code is correct. If you think you may be affected, replace `nn/base.py` with the new version from assignment2.zip.

Update (4/27): A new version of the handout PDF is available, which fixes a few typos and adds some clarifications. See Piazza for more details.

In this assignment you will become familiar with deep feed-forward neural networks and with recurrent neural networks (RNNs), and apply them to the tasks of Named Entity Recognition (NER) and Language Modeling (LM).

As with Assignment #1, you're limited to a maximum of three late days on this assigment. Don't forget that the in-class midterm is scheduled for May 6, so we recommend starting this one early!

Setup

Note: *Please be sure you have Python 2.7.x installed on your system, and follow the setup instructions below.*

Get the code (updated!): Download the starter code here and the assignment handout here..

Python package requirements: We'll be using IPython (Jupyter) notebooks in this assignment just like the last one. If you are able to open those, then the notebooks for this assignment should work fine. You'll also need the following packages, which may already be installed:

- numpy 1.9.2
- matplotlib 1.4.3
- scikit-learn (sklearn) 0.16.0
- pandas 0.15.2

If you had trouble setting up IPython or other dependencies for the last assignment, or if you are using Windows, we recommend installing the Anaconda Python distribution (use the Python 2.7 version). It includes a package manager `conda` that greatly simplifies installing and linking most common Python packages.

Additionally, if you have an Intel CPU you can speed up your models by using Intel MKL - this will automatically parallelize many matrix and vector operations in NumPy. The easiest way to get this is to install a compatible distribution: Anaconda and Enthought Canopy both include this in their full versions, which are free for academic use.

If you use Anaconda: just run

```
conda update conda
conda update anaconda
conda update scikit-learn
```

and you should be good to go.

If you set up a `virtualenv` for the last assignment, then do the following, as with Assignment #1:

```
cd assignment2
sudo pip install virtualenv      # This may already be installed
virtualenv .env                  # Create a virtual environment
source .env/bin/activate         # Activate the virtual environment
pip install -r requirements.txt  # Install dependencies
# Work on the assignment for a while ...
deactivate                       # Exit the virtual environment
```

If you used `pip` (e.g. `pip install -r requirements.txt` or otherwise) on the last assignment without a `virtualenv`, then you should

```
cd assignment2
```

```
pip install -r requirements.txt
```

Submitting your work

This assignment requires you to submit a number of files, which are listed in the assignment handout under **"deliverables"** for each section. Please be sure to use the provided code snippets in the notebook to generate these files, and follow the instructions below to submit a .zip file and your writeup PDF.

Submission Logistics

There are four steps to submit your assignment, so ***please read the following carefully:***

1. Copy Part 1.1 code to `part11probing.py`
2. Run the sanity checker `collectSubmission.py`
3. Upload `<your-sunet-id>.zip` to Box
4. Upload your writeup PDF to Scoryst

Step 1: In order to evaluate Part 1.1 (Probing Neuron Responses), we ask that you copy-and-paste your code - minus the print statements - for 1.1(a), 1.1(b), and 1.1(c) from the `part1-NER.ipynb` notebook into the respective functions `part_a()`, `part_b()`, and `part_c()` in the template file `part11probing.py` which you can download here. Put this file in your assignment directory.

Unless you made significant modifications to the starter code (*not recommended*), this should only take a couple minutes. You can test your code by running, in the notebook:

```
from part11probing import part_a, part_b, part_c
# clf is your trained WindowMLP model
part_a(clf, num_to_word)
part_b(clf, num_to_word, num_to_tag)
part_c(clf, num_to_word, num_to_tag)
```

or alternatively by running `python part11probing.py` from the command line to test your code on a nonsense model.

Step 2: We've written a sanity checker script that will run a few *very basic* tests to make

sure your output and code are in the correct format. Please note that this script *does not* verify correctness or guarantee that all output is in the correct format. *It only checks for a few cases of malformatted output* and *does not run gradient checks*! The script will also check for all the files you need to submit, and will prepare a .zip file for submission. This script should work on all platforms (Mac, Linux, Windows).

Download `collectSubmission.py` here, and put it in your assignment directory. Then run as:

```
python collectSubmission.py
```

The script will warn you if files are missing or tests fail; you may opt to continue and it will still prepare the .zip, but you do so at your own risk.

Step 3: Upload `<your-sunet-id>.zip`, as output by the sanity checker, to the Box folder for this assignment

Extra Credit: If you implemented the `ExtraCreditRNNLM` extensions, please upload a separate zip file containing any additional parameter files (i.e. `extracredit.U.npy`, etc.) necessary to test your model. Name this file `<your-sunet-id>-extracredit.zip`, and upload it to the same Box folder as the rest of the assignment.

Step 4: Upload your writeup PDF to Scoryst, under Assignment 2. Unfortunately, Scoryst is 1-indexed and fairly inflexible about numbering, so please note the following:

- Part 0 (XOR) is "Question 1"
- Part 1 (NER) is "Question 2"
- Part 2 (RNNLM) is "Question 3"
- For questions (such as 1(d) a.k.a. "Question 2 part 4") with both programming *and* written components, point Scoryst to the corresponding part of your writeup. We'll grade your code and include it in these parts.
- For programming-only questions (such as 0(c) a.k.a "Question 1 part 3"), just point Scoryst to a dummy location; we'll enter the code grade as in Assignment 1.
- Part 2 (RNNLM): "Question 3", parts 8 and 9 are for Extra Credit: unigram filling and RNNLM optimizations/extensions, respectively. If you did these, please point these questions to the relevant section of the report.

Basically, if you submit everything ***in the order it appears*** you should be OK.

Assignment Overview (Tasks)

There are three parts to this assignment, each of which has both written and programming components. As with Assignment #1, the programming components are centered around IPython notebooks. To reduce clutter and scrolling, we've given you three separate notebooks, `part0-XOR`, `part1-NER`, and `part2-RNNLM`, corresponding to each part of the assignment. You'll also be writing code in standalone python files `misc.py`, `nerwindow.py`, and `rnnlm.py` to implement your models.

Be sure to follow the instructions carefully, in the handout, notebook, and comments that define the functions you'll be implementing. As with Assignment #1, we strongly recommend that you complete the written component for each section before you beginning the accompanying programming section.

Part 0: Boolean Logic (10 points)

Part 1: NER Window Model (45 points)

- (a),(b),(c): 15 points
- (d),(e),(f) and (1.1): 30 points

Part 2: RNN Language Model (45 points)

- (a),(b),(c),(d): 20 points
- (e),(f),(g): 25 points