11-785 Spring 2018 Homework 3 Part 1

February 28, 2018

1 Introduction

In part 1 of homework 3, we will be training a recurrent neural network on the WikiText-2 language modeling dataset. You will practice using a recurrent network to model and generate text. You will also learn about the various techniques we use to regularize recurrent networks and improve their performance.

The below sections will describe the dataset and what your model is expected to do. You are responsible at this point for organizing your training as you see fit.

Please refer to Regularizing and Optimizing LSTM Language Models for information on how to properly construct, train, and regularize an LSTM language model `https://arxiv.org/pdf/1708.02182.pdf`. You are not expected to implement every method in that paper. Our tests are not overly strict and you will have performance sufficient to pass Autolab with only a subset of those methods.

These tests require that you train the model on your own and submit a tarball containing the model as well as code capable of running the model. Details follow below.

1.1 Autograder Submission

Your solutions will be autograded by Autolab. For this reason, it is important that you do not change the signature of any of the functions contained in the template.

In order to submit your solution, create a tar file containing your code. The root of the tar file should have a directory named hw3 containing your module code.

1.2 PEP8 (10 points)

Format your code according to PEP8 guidelines. You might not receive help on poorly formatted code. Modern IDEs should have a shortcut to automatically reformat your code.

In order to check your code for PEP8, install pytest and pytest-pep8 and run the following command.

```
pip install pytest pytest-pep8
pytest --pep8 -m pep8 [path-to-your-code]
```

The autograder will check all PEP8 guidelines except that the maximum line length will be raised to 120.

2 Dataset

A preprocessed WikiText-2 dataset is included in the template tarball.

- `vocab.npy` a numpy file containing the words in the vocabulary
- `vocab.csv` a human-readable CSV file listing the vocabulary
- `wiki.train.npy` a numpy file containing training text
- `wiki.valid.npy` a numpy file containing validation text

The vocabulary file contains an array of strings. Each string is a word in the vocabulary. There are 33,278 vocabulary items.

The train and validation file contain an array of articles. Each article is an array of integers, corresponding to words in the vocabulary. There are 579 articles in the training set.

For example, the first article in the training set contains 3803 integers.

The first 6 integers of the first article are `[1420 13859 3714 7036 1420 1417]`.

Looking up these integers in the vocabulary reveals the first line: `= Valkyria Chronicles III = <eol>`.

3 Training

You are free to structure the training and engineering of your model as you see fit. Follow the protocols in the paper as closely as you are able in order to guarantee maximal performance.

The following regularization techniques will be sufficient to achieve performance to pass on Autolab. Refer to the paper for additional details and please ask for clarification on Piazza. It may not be necessary to utilize all of the below techniques, depending on your luck in training the network.

- Apply locked dropout between LSTM layers
- Apply embedding dropout
- Apply weight decay
- Tie the weights of the embedding and the output layer

- Activity regularization
- Temporal activity regularization

Your model will likely take around 15-20 epochs, approximately 30 minutes on a GPU, to achieve a validation NLL below 5.5. The Autolab tests require a performance of 5.5. Performance reported in the paper is 4.18, so you have room for error.

Data is provided as a collection of articles. You may concatenate those articles to perform batching as described in the paper. It is advised to shuffle articles between epochs if you take this approach.

4 Prediction of a Single Word (45 points)

Complete the function `prediction` in the file `prediction.py`.

This function takes as input a batch of sequences, shaped `[batch size, sequence length]`.

This function should load your trained model and perform a forward pass. Return the scores for the next word after the provided sequence for each sequence. The returned array should be `[batch size, vocabulary size]` (`float`).

These input sequences will be drawn from the unseen test data. Your model will be evaluated based on the score it assigns to the actual next word in the test data.

Note that scores should be raw linear output values. Do not apply softmax activation to the scores you return.

Required performance is a negative log likelihood of 5.5. The value reported by autolab for the test dataset should be similar to the validation NLL you calculate on the validation dataset. If these values differ greatly, you likely have a bug in your code.

5 Generation of a Sequence (45 points)

Complete the function `generation` in the file `generation.py`.

As before, this function takes as input a batch of sequences, shaped `[batch size, sequence length]`.

Instead of only scoring the next word, this function should generate an entire sequence of words. The length of the sequence you should generate is provided in the `forward` parameter. The returned shape should be `[batch size, forward]` (`int`).

This function requires sampling the output at one timestep and using that as the input at the next timestep. Please refer to reciation 6 for additional details on how to perform this operation.

Your predicted sequences will be passed through a model that we have trained and the NLL of your outputs will be calculated. If your outputs make

sense, they will have a reasonable NLL. If your outputs do not reasonably follow the given outputs, the NLL will be poor.

In other words, we trained a model to recognize Wikipedia articles. If you generate text that looks like a Wikipedia article to our model, then you are generating data correctly.

6 Local Testing

You may run tests locally using the command `pytest tests`.

Note that evaluating your sequence generation requires a trained model, so we cannot provide this in the template. The template contains a test that will run your model and print the generated text. If that generated text seems like English, then the test on Autolab will likely pass.

Good Luck and Have Fun!