LING 575K HW7

Due 11PM on May 20, 2021

In this assignment, you will

- Develop understanding of recurrent neural networks, especially as used for language modeling
- Implement components of data processing
- Implement masking of losses for an RNN language model

All files referenced herein may be found in `/dropbox/20-21/575k/hw7/` on patas.

1 Recurrent Neural Network Decoders/Taggers

Q1: Understanding RNNs

Q2:

2 Implementing an RNN Character Language Model

In the coding portion of this assignment, you will implement (components of) an LSTM character-level language model for the Stanford Sentiment Treebank, using an RNN as tagger.

Q1: Data processing The reviews in the SST dataset come in various lengths. In the previous models we have looked at in the class, this has not been an issue because they rely either on a bag-of-words representation (Deep Averaging Network) or a fixed-sized window of previous tokens (Feed-Forward Language Model). RNNs, however, require the use of *padding*: given a batch of reviews of various lengths, we pad the shorter sequences with a special padding token so that all sequences are as long as the longest one.
In `data.py`, please implement the `pad_batch` method. Please read the method signature and docstring carefully for details on the input and output.

Q2: Vanilla RNN Cell The "cell" of an RNN does one time-step of computation. For a Vanilla RNN, we saw that this was

$$h_t = \tanh\left(W_h h_{t-1} + b_h + W_x x_t + b_x\right)$$

where h_{t-1} is the previous hidden state, x_t is the current input, and the Ws and bs are parameters for linear transformations.
In `model.py`, implement this computation in `VanillaRNNCell.forward`. The initializer defines the linear layers that you will need.

Q3: LSTM Cell An LSTM cell computes the next hidden state and *memory* based on the previous hidden state and memory together with the current input. Please consult these slides for the entire set of equations (and details about motivation).

In `model.py`, implement this computation in `LSTMCell.forward`. The initializer defines the linear layers that you will need.

3 Running the Language Model

`run.py` contains a basic training loop for SST language modeling. It will record the training and dev loss (and perplexity) at each epoch, and save the best model according to dev loss. Periodically (as specified by a command-line flag), it also outputs generated text from the best model.

Q1: Default parameters

Q2: Modify hyper-parameterr

4 Testing your code

In the dropbox folder for this assignment, we will include a file `test_all.py` with a few very simple unit tests for the methods that you need to implement. You can verify that your code passes the tests by running `pytest` from your code's directory, with the course's conda environment activated.

Submission Instructions

In your submission, include the following:

- readme.(txt|pdf) that includes your answers to §1 and §3.
- `hw7.tar.gz` containing:
  - run_hw7.sh. This should contain the code for activating the conda environment and your run commands for §3 above. You can use run_hw2.sh from the previous assignment as a template.
  - data.py
  - run.py