Programming Assignment 3: RNN Language Model[1]

Deadline: Wednesday, March 22, at 11:59pm
TA: Yuhuai (Tony) Wu (`csc321ta@cs.toronto.edu`)

Submission: You must submit two files through MarkUs[2]: a PDF file containing your writeup, titled `a3-writeup.pdf`, and your code file `a3_code.py`. Your writeup must be typeset using LaTeX.

The programming assignments are individual work. See the Course Information handout[3] for detailed policies.

You should attempt all questions for this assignment. Most of them can be answered at least partially even if you were unable to finish earlier questions. If you were unable to run the experiments, please discuss what outcomes you might hypothetically expect from the experiments. If you think your computational results are incorrect, please say so; that may help you get partial credit.

Overview

In thie project, you will work on extending `min-char-rnn.py`, the vanilla RNN language model implementation we covered in tutorial. This was written by Andrej Karpathy[4]. You will experiment with the Shakespeare dataset, which is `shakespeare.txt` in the starter code.

Part 1 (20%)

The RNN language model uses a softmax activation function for its output distribution at each time step. It's possible to modify the distribution by multiplying the logits by a constant α:

$$\mathbf{y} = \text{softmax}(\alpha \mathbf{z}).$$

Here, $1/\alpha$ can be thought of as a "temperature", i.e. lower values of α correspond to a "hotter" distribution. (This terminology comes from an algorithm called simulated annealing.)

Write a function to sample text from the model using different temperatures (i.e., 1/alpha on Slide 5). Try different temperatures, and, in your report, include examples of texts generated using different temperatures. Briefly discuss what difference the temperature makes.

Include the listing (i.e., source code) of the function you wrote/modified to accomplish the task in the report.

You should either train the RNN yourself, or use the weights from Part 3 — up to you.

Part 2 (50%)

Write a function that uses an RNN to *complete* a string. That is, the RNN should generate text that is a plausible continuation of a given starter string. In order to do that, you will need to

[1] This assignment was originally written by Renjie Liao and Michael Guerzhoy.
[2] `https://markus.teach.cs.toronto.edu/csc321-2017-01`
[3] `http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/syllabus.pdf`
[4] `https://gist.github.com/karpathy/d4dee566867f8291f086`

compute the hidden activity **h** at the end of the starter string, and then to start generating new text.

Include 5 interesting examples of outputs that your network generated using a starter string. (This part need not be easily reproducible).

Include the listing (i.e., source code) of the function you wrote in the report.

You should either train the RNN yourself, or use the weights from Part 3 – up to you.

Part 3 (30%)

The weights for a trained RNN are included as `char-rnn-snapshot.npz`. Some samples from the RNN (at temperature $1/\alpha = 1$) are included as `samples.txt`, and code to read in the weights is included as `read_in_npz.py` (if this doesn't work, try the pickle file, and get a using `import cPickle as pickle; a = pickle.load(open("char-rnn-snapshot.pkl"))`.)

In the samples that the RNN generated, it seems that a newline or a space usually follows the colon (i.e., ":") character. In the weight data provided, identify the specific weights that are responsible for this behaviour by the RNN. In your report, specify the coordinates and values of the weights you identified, and explain how those weights make the RNN generate newlines and spaces after colons.

Part 4 (10% bonus)

Identify another interesting behaviour of the RNN, identify the weights that are responsible for it. Specify the coordinates and the values of the weights, and explain how those weights lead to the behaviour that you identified. To earn bonus points, the behaviour has to be more interesting than the behaviour in Part 3 (i.e., character A following character B).

What to submit

The project should be implemented using Python 2. Your report should be in PDF format, generated with LaTeX. You will submit three files: `a3_code.py`, and `a3-writeup.pdf`. You can submit other Python files as well: we should have all the code that's needed to run your experiments.

Reproducibility counts! We should be able to obtain all the graphs and figures in your report by running your code. Submissions that are not reproducible will not receive full marks. If your graphs/reported numbers cannot be reproduced by running the code, you may be docked up to 20%. (Of course, if the code is simply incomplete, you may lose even more.) Suggestion: if you are using randomness anywhere, use `numpy.random.seed()`.