CS4803/7643: Deep Learning
Spring 2020
Problem Set 3

Instructor: Zsolt Kira
TAs: Yihao Chen, Sameer Dharur, Rahul Duggal, Patrick Grady, Harish Kamath
Yinquan Lu, Anishi Mehta, Manas Sahni, Jiachen Yang, Zhuoran Yu
Discussions: https://piazza.com/gatech/spring2020/cs4803dl7643a

Due: Tuesday March 15, 2020, 11:59pm

Instructions

1. We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   - Each subproblem must be submitted on a separate page. When submitting to Gradescope, make sure to mark which page(s) corresponds to each problem/sub-problem
   - Remember to append your notebook PDFs to the solutions for the problem set, as described in the instructions!
   - For the coding problem, please use the provided collect_submission.sh script and upload cs7643_hw3.zip to the HW3 Code assignment on Gradescope. While we will not be explicitly grading your code, you are still required to submit it. Please make sure you have saved the most recent version of your jupyter notebook before running this script.
   - Note: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions. Please read https://stats200.stanford.edu/gradescope_tips.pdf for additional information on submitting to Gradescope.

2. LaTeX'd solutions are strongly encouraged (solution template available at cc.gatech.edu/classes/AY2020/cs7643_spring/assets/sol2.tex), but scanned handwritten copies are acceptable. Hard copies are **not** accepted.

3. We generally encourage you to collaborate with other students.

   You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and *not* as a group activity. Please list the students you collaborated with.

1 Recurrent Neural Network

1. **[Vanilla RNN for parity function: 4 points]**

   Let us define a *sequence parity function* as a function that takes in a sequence of binary inputs and returns a sequence indicating the number of 1's in the input so far; specifically, if at time t the 1's in the input so far is even it returns 1, and 0 if it is odd. For example, given input sequence $[0, 1, 0, 1, 1, 0]$, the parity sequence is $[1, 0, 0, 1, 0, 0]$. Let x_t denote the input at time t and y_t be the boolean output of this parity function at time t.

   Design a vanilla recurrent neural network to implement the parity function. Your implementation should include the equations of your hidden units and the details about activations with different input at x_t (0 or 1).

   Hint: Recall that we have implemented AND and OR functions in problem set 2, which may be useful here.

2. **[LSTM for parity function: 4 points]**

   Let us recall different gates in an LSTM Network. First gate is the "forget gate layer":

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{1}$$

   where f_t is the output of forget gate, W_f is the weight matrix, h_{t-1} is the hidden state of step *t-1*, x_t is the current input and b_t is the bias value.
   Next we have "input gate layer":

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{3}$$

   where i_t decides which values we will update and $\tilde{C}_t$ are the new candidate values that could be added to the cell state. Next we have new cell state candidate values:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

   Finally, we have the output and hidden state

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * \tanh(C_t) \tag{6}$$

   Design an LSTM Network for the bit parity problem mentioned in Question 1. Specifically, provide values for W_f, b_f, W_i, b_i, W_C, b_C, W_o and b_o such that the cell state C_t store the parity of bit string. Please mention any assumptions you make. For this problem, you can assume below for Sigmoid and tanh function:

$$\sigma(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$\tanh(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & x = 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{8}$$

Hint: Recall that XOR of x and y can be represented as $(x \wedge \bar{y}) \vee (\bar{x} \wedge y)$. Think about how you can leverage this information for equation (4).

3. **[When to stop in beam search?: 4 points]**

Beam Search is a technique widely used to improve the output text quality of neural text generation. But it is difficult to decide when to stop beam search to obtain optimality because hypotheses can finish in different steps. Assume an input $\mathbf{x}$ from which we generate an output sentence $\mathbf{y}$ which is a *completed* hypothesis:

$$\mathbf{y}* = \underset{\mathbf{y}:comp(\mathbf{y})}{\mathbf{argmax}} \prod_{i \leq |y|} p(y_i|x, y_{<i}) \tag{9}$$

where $\mathbf{y}_{<i}$ is a shorthand notation of $y_0 y_1 ... y_{i-1}$ and $comp(\mathbf{y})$ is shorthand for completed hypothesis. We say that a hypothesis $\mathbf{y}$ is **completed** ($comp(\mathbf{y})$), if its last word is `</s>`, i.e.,

$$comp(\mathbf{y}) \stackrel{\text{def}}{=} (\mathbf{y}_{|\mathbf{y}|} = \texttt{</s>})$$

in which case it will not be further expanded.

Given B_{i-1} is an b length ordered list, consisting of pairs $\langle \mathbf{y}, s \rangle$ of the current prefix $\mathbf{y}$ and the accumulated score (product of probabilities) s , we use beam search to approximately find the best output $\mathbf{y}^*$. Beam search can be defined as an operator $\mathbf{top}^b$ that selects the top b scoring items:

$$B_i = \overset{b}{\mathbf{top}}\{\langle \mathbf{y}' \circ y_i,\ s \cdot p(y_i|\mathbf{x}, \mathbf{y}) \rangle \mid \langle \mathbf{y}', s \rangle \in B_{i-1}\} \tag{10}$$

Let us define a beam search algorithm that will always return the optimal score complete hypothesis (modulo beam size) and finish as soon as optimality is reached. Let $best_{\leq i}$ be the best completed hypothesis so far up to step i, i.e.

$$best_{\leq i} \stackrel{\Delta}{=} max\Big\{\mathbf{y} \in \cup_{j \leq i} B_j | comp(\mathbf{y})\Big\} \tag{11}$$

We update it every time we find a completed hypothesis (if there is none yet, then it remains undefined). Given that at any step i, if $best_{\leq i}$ is defined and the highest scoring item in the current beam B_i scores worse than or equal to $best_{\leq i}$, prove that the current best completed hypothesis (obtained from $best_{\leq i}$) is the overall highest-probability completed hypothesis and future steps will be no better than the best completed hypothesis.

4. **[Exploding Gradients: 4 points; Extra credit for 4803]**

Learning long-term dependencies in recurrent networks suffers from a particular numerical challenge – gradients propagated over many time-steps tend to either 'vanish' (i.e. converge to 0, frequently) or 'explode' (i.e. diverge to infinity; rarely, but with more damage to the optimization). To study this problem in a simple setting, consider the following recurrence relation without any nonlinear activation function or input x:

$$h_t = W^\top h_{t-1} \tag{12}$$

where W is a weight sharing matrix for recurrent relation at any time t. Let $\lambda_1, ..., \lambda_n$ be the eigenvalues of the weight matrix $W \in \mathbb{C}^{n \times n}$. Its spectral radius $\rho(W)$ is defined as:

$$\rho(W) = \max\{|\lambda_1|, ..., |\lambda_n|\} \tag{13}$$

Assuming the initial hidden state is h_0, write the relation between h_T and h_0 and explain the role of the eigenvalues of W in determining the 'vanishing' or 'exploding' property as $T \gg 0$

5. **[Paper Review: 4 points]**

   Explainability is an increasingly important area of deep learning research today. The following paper, from CVPR 2018, introduces a unique multi-modal approach to explainability by joint textual rationale generation and attention visualization in the task of visual question answering (VQA). This is a major improvement over pre-existing unimodal explainable models, with the results showing complementary explanatory strengths from the visual and textual explanations. The paper can be accessed here.

   As in the previous assignments, please limit your reviews to 350 words.

   Briefly summarize the key findings, strengths and potential limitations of this work.

2 Coding: Sequence models for image captioning [55 regular points for CS7643, 45 regular points for CS4803]

The coding part of this assignment will consist of implementation of sequence models for captioning images. To get started, go to https://www.cc.gatech.edu/classes/AY2020/cs7643_spring/Z3o9P26CwTPZZMDXyWYDj3/hw3/