Homework 4 — Recurrent Neural Networks

1 Scope

This assignment will involve using recurrent neural networks in natural language processing applications.

2 Instructions

- **18-780 students** will only need to complete Deliverables 2 and 3. Their deadline is March 1st. We will leave the portal open till Tuesday March 5th for any late submission.
- **18-786 students** will need to complete all deliverables. Their deadline is March 15th.
- Your submission has two parts: (i) a PDF report that details all the requested deliverables and (ii) the code for the assignment. You will upload the PDF report in gradescope, marking which parts of the report corresponds to which deliverable. The code will be uploaded separately as a collection of .py files, and we will run "software similarity" software on it to detect violations of the integrity policy.
- You can discuss this HW with others, but you are **not** allowed to share, borrow, copy or look at each other's codes.
- All code that you submit must be yours (except for the starter code that we provide).

3 Problem Set

Recurrent neural networks (RNNs) are designed to process sequential data, such as those found in language, speech, and economics. For this assignment, we will focus on language.

(Deliverable 1: Unstable RNNs) *This deliverable is only required for students in 18-786.* Define linear layers $f(x) = Ax + b$ and $g(x) = Cx + d$. For an input x_0 and activation σ, let our RNN have the following simple architecture:

$$x_t = \sigma(f(x_{t-1})) \tag{1}$$
$$y_t = g(x_t). \tag{2}$$

We want to show that these simple RNNs are prone to producing rubbish. For simplicity, let $A = C = \begin{bmatrix} 1.25 & 0.75 \\ 0.75 & 1.25 \end{bmatrix}$ and $b = d = 0$. First, sample 10 vectors from the standard normal distribution which will be used as $x_0 \in \mathbb{R}^2$.

Repeat the following for activation functions $\sigma(x) = x$, $\sigma(x) = ReLU(x)$, and $\sigma(x) = tanh(x)$, including your results into your report:

1. Plot the relationship between t and $\|y_t\|_2$ for $t = 0, \ldots, 15$ for your 10 sampled x_0's. All 10 trajectories should be in the same plot.

2. Plot the relationship between t and $\|y_t\|_2$ for $t = 0, \ldots, 15$ for $x_0 = \begin{bmatrix}1\\1\end{bmatrix}$ and $x_0 = \begin{bmatrix}1\\-1\end{bmatrix}$. Both trajectories should be labeled and shown in the same plot.

Additionally, summarize your findings in your report. Include the code in your submission as `deliverable1.py`. *No starter code will be given for this part.* Note that you do not need to implement the backward pass of the RNN.

(Deliverable 2: Evaluation Metrics) An immediate challenge we face with language is turning them into numeric data. One way to do this is a process known as *tokenization*, where words, subwords, and/or phrases are assigned a nonnegative integer (tokens). This way, a string of text is converted to a list of numbers. For instance, suppose our tokenizer is defined with the following vocabulary dictionary: {"be": 0, "to": 1, ",": 2, "or": 3, "not": 4}. Then, the sentence "to be, or not to be" would be tokenized into [1, 0, 2, 3, 4, 1, 0]. Now, we can train our language models by inputting one-hot encodings and minimizing cross entropy loss by taking each unique token as a class!

For many language tasks, evaluating a model's performance in an interpretable manner is not straightforward. For instance, how can you quantify the quality of a translation? Or how about the quality of summarized text? Maybe you can do it by hand, but how would you automate it? Consequently, there has been research to develop "good" metrics for language tasks. We explore some of them here.

- Accuracy: classification error
- Perplexity (PPL): the exponential of the mean cross entropy of the entire dataset
- BLEU score: The amount of overlapping n-grams between the model output and true text. An n-gram is a length-n sequence of consecutive tokens. This metric also has a penalty for overly short outputs. Discussed more below.

BLEU scores. (This section is adapted from a homework in 10-707.) Let's first discuss n-grams in more detail. An n-gram is a length-n sequence of consecutive tokens. So, for a sentence [Sam, likes, to, eat, apples], the 1-grams are [(Sam), (likes), (to), (eat), (apples)], and the 3-grams are [(Sam, likes, to), (likes, to, eat), (to, eat, apples)]. Finding n-grams is the first step to calculating BLEU scores.

Suppose y is our length-s target sequence, and $\hat{y}$ is the model's length-$\hat{s}$ output. Fixing n, for each *unique* n-gram in $\hat{y}$, find the minimum between the number of times an n-gram appears in y and the number of times it appears in $\hat{y}$. Summing these up and scaling by the number of n-grams in $\hat{y}$, $\hat{s} - n + 1$, gives us the the clipped precision score for n:

$$P_n(y, \hat{y}) = \frac{\sum_{g \in \{\text{unique } n\text{-grams in } \hat{y}\}} \min(C(\hat{y}, g), C(y, g))}{\hat{s} - n + 1} \tag{3}$$

where $C(y, g)$ is number of times the n-gram g appears in y and similarly for $C(\hat{y}, g)$.

Next, we find the geometric mean of all $P_n(y, \hat{y})$ for n up to k: $\prod_{n=1}^{k} P_n(y, \hat{y})^{1/k}$. Finally, we need a brevity penalty to penalize outputs which are too short by scaling this product with $\min(1, \exp(1 - \frac{s}{\hat{s}}))$. (Think what happens if $\hat{y}$ is a single word that appears in y.). Putting it together, the BLEU-k

score is

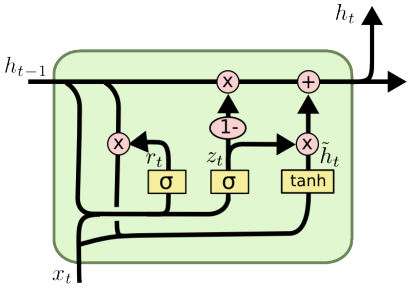
$$\min(1, \exp(1 - \frac{s}{\hat{s}})) \prod_{n=1}^{k} P_n(y, \hat{y})^{1/k}. \tag{4}$$

Please answer the following questions in your report:

1. Suppose we want the model to perform *single-word sentence completion* (e.g. without context, I ask a model to fill in the blank of the following sentence with a single word: "I like to eat apple _____ after dinner."). Which evaluation metric (accuracy, PPL, or BLEU) would be appropriate and why?
2. Suppose we want the model to *summarize an article*. Which evaluation metric (accuracy, PPL, or BLEU) would be appropriate and why?
3. Suppose we want the model to perform *sentiment analysis*, the task of determining whether a sentence has a positive or negative sentiment (e.g. "This is Sam's favorite movie." has positive sentiment). Which evaluation metric (accuracy, PPL, or BLEU) would be appropriate and why?
4. Suppose we want the model to perform *multiple choice question answering*. Which evaluation metric (accuracy, PPL, or BLEU) would be appropriate and why?
5. What is a weakness of BLEU scores? *Hint: How would BLEU and a human evaluate the similarity of these sentences: "She received the highest score." (target) vs. "No one got more points than her." (output)*

For your code submission, please implement BLEU-k score in `deliverable2.py`. This will be used in the next problem. Several test cases are given to you in `test_bleu.py` to check your implementation. Simply run `python test_bleu.py`, and if all tests pass, nothing will be returned.

(Deliverable 3: Translation) *This problem requires training for approximately 1 hr on CPU, so start early!*



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 1: GRU Architecture

In this problem, we will use Gated Recurrent Units (GRUs), a simplified version of LSTMs, for translation. The given notebook, `deliverable3.ipynb`, (adapted from https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html) can be *nearly* run from start to finish as is — no model or hyperparameter adjustments necessary. Therefore, please read through each block in the notebook carefully while you run them, as part of the deliverable will be answering specific questions about the implementation.

Your tasks are the following:

1. To check your understanding of the model, please answer the following in your report:
   (a) How are words encoded as vectors?
   (b) What is the maximum sentence length that is used to train this model?
   (c) What do we feed into the encoder?
   (d) What is produced by the decoder?
   (e) How many samples are in the original and trimmed datasets?
   (f) How are sentences processed before feeding them into the our BLEU-k score implementation?
2. Use your implementation of BLEU score to evaluate the performance of the model on the given evaluation dataset. Include in your report the BLEU-k scores for $k = 1, 2, 3, 4, 5$ rounded to 4 decimal places. Do you notice a trend?
3. Include in your report 5 example triples of inputs, targets, and model outputs.