COM S 673: Advanced Topics in Computational Models of Learning
Assignment #3 (part b)

1. You need to submit a report and your code to Canvas. Your hard-copy report should include (1) answers to the non-programming part, and (2) analysis and results of the programming part. Please put all your code files and report into a compressed file named "HW#_FirstName_LastName.zip"
2. Unlimited number of submissions are allowed and the latest one will be timed and graded.
3. Please read and follow submission instructions. No exception will be made to accommodate incorrectly submitted files/reports.
4. All students are required to typeset their reports using latex.
5. Only write your code between the following lines. Do not modify other parts.

   ### YOUR CODE HERE

   ### END YOUR CODE

Please choose A or B for this assignment. The total points for this assignment is 40. If you choose B, then the maximum points you can get is 60 (20 points are extra points).

1. Option A (40 points) In this assignment, you will implement a recurrent neural network (RNN) for language modeling using PyTorch. The task is to predict word x_{t+1} given words $x_1, \dots, x_t$:

$$P(x_{t+1} = v_j | x_t, \dots, x_1)$$

where v_j is the j-th word in the vocabulary. The file"utils.py" gives an example of how to generate the vocabulary. You can read it if interested. With the vocabulary, we can transform a word x_i into a one-hot vector. Suppose our RNN model is, for $t = 1, \dots, n-1$:

$$\begin{aligned} e^{(t)} &= x^{(t)}L, \\ h^{(t)} &= \text{sigmoid}(h^{(t-1)}H + e^{(t)}I + b_1), \\ \hat{y}^{(t)} &= \text{softmax}(h^{(t)}U + b_2), \\ \bar{P}(x_{t+1} &= v_j | x_t, \dots, x_1) = \hat{y}_j^{(t)}. \end{aligned}$$

where the first line actually corresponds to a word embedding lookup operation. $h^{(0)}$ is the initial hidden state, $\hat{y}^{(t)} \in \mathbb{R}^{|V|}$ and its j-th entry is $\hat{y}_j^{(t)}$.

Training parameters θ in this model are:

· L – embedding matrix which transforms words in the vocabulary into lower dimensional word embedding.

· H – hidden transformation matrix.

· I – input transformation matrix which takes word embedding as input.

· U – output transformation matrix which projects hidden state into prediction vector.

- b_1 – bias for recurrent layer.
- b_2 – bias for projection layer.

(a) (10 points) Let the dimension of word embedding as d, the size of vocabulary as $|V|$, the number of hidden units as D, please provide the size of each training parameter above.
Answer. Please note, the expression $(\hat{y}^{(t)} - y^{(t)})U^\top \odot h^{(t)} \cdot (1 - h^{(t)})$ is computed element-wise, where each component is $\left[(\hat{y}^{(t)} - y^{(t)})U^\top\right]_j \cdot h_j^{(t)} \cdot (1 - h_j^{(t)})$ for $j = 1, \ldots, D$.

The embedding lookup operation is given by:

$$e^{(t)} = x^{(t)} L, \quad \text{where } x^{(t)} \in \mathbb{R}^{1\times|V|},\ L \in \mathbb{R}^{|V|\times d},\ e^{(t)} \in \mathbb{R}^{1\times d}.$$

This operation retrieves the word embedding vector $e^{(t)}$ for the input word $x^{(t)}$, which is represented as a one-hot vector over the vocabulary.
We are also given that the output prediction $\hat{y}^{(t)} \in \mathbb{R}^{1\times|V|}$. Since:

$$\hat{y}^{(t)} = \text{softmax}(h^{(t)} U + b_2),$$

it follows that:

$$h^{(t)} U \in \mathbb{R}^{1\times|V|}, \quad \text{and} \quad b_2 \in \mathbb{R}^{1\times|V|}.$$

Therefore, $h^{(t)} \in \mathbb{R}^{1\times D}$ and $U \in \mathbb{R}^{D\times|V|}$, as U projects the hidden state into the vocabulary space.
The hidden state update is defined as:

$$h^{(t)} = \text{sigmoid}(h^{(t-1)} H + e^{(t)} I + b_1),$$

where:

- $h^{(t-1)} H \in \mathbb{R}^{1\times D}$ implies $H \in \mathbb{R}^{D\times D}$,
- $e^{(t)} I \in \mathbb{R}^{1\times D}$ implies $I \in \mathbb{R}^{d\times D}$,
- and $b_1 \in \mathbb{R}^{1\times D}$.

The following table summarizes the dimensions of each parameter and intermediate variable:

Parameter	$e^{(t)}$	$x^{(t)}$	L	$\hat{y}^{(t)}$	$h^{(t)}$	U	b_1	b_2	H	I
Shape	$\mathbb{R}^{1\times d}$	$\mathbb{R}^{1\times\|V\|}$	$\mathbb{R}^{\|V\|\times d}$	$\mathbb{R}^{1\times\|V\|}$	$\mathbb{R}^{1\times D}$	$\mathbb{R}^{D\times\|V\|}$	$\mathbb{R}^{1\times D}$	$\mathbb{R}^{1\times\|V\|}$	$\mathbb{R}^{D\times D}$	$\mathbb{R}^{d\times D}$

Table 1: Dimensions of model parameters and variables in the RNN language model.

(b) (20 points) To train the model, we use *cross-entropy* loss. For time step t (note that for language model, we have loss for every time step), we have:

$$E^{(t)}(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = -\sum_{j=1}^{|V|} y_j^{(t)} \log(\hat{y}_j^{(t)}),$$

where $y^{(t)}$ is the one-hot vector corresponding to the target word, which here is equal to $x^{(t+1)}$. For a sequence, we sum the loss of every time step.

Modern deep learning libraries like **PyTorch** does not require the implementation of back-propagation, but you should know the details. Compute the following gradients at a single time step t:

$$\frac{\partial E^{(t)}}{\partial U}, \frac{\partial E^{(t)}}{\partial b_2}, \frac{\partial E^{(t)}}{\partial I}|_{(t)}, \frac{\partial E^{(t)}}{\partial H}|_{(t)}, \frac{\partial E^{(t)}}{\partial b_1}|_{(t)}, \frac{\partial E^{(t)}}{\partial h^{(t-1)}}$$

where $|_{(t)}$ denotes the gradient with respect to time step t only. Note that we have weight sharing in recurrent layer, so in practice, the back-propagation would update the parameters according to the gradients across all time steps. Hint:

$$\frac{\partial E^{(t)}}{\partial U} = (h^{(t)})^T(\hat{y}^{(t)} - y^{(t)})$$

Make sure you understand this hint and you can use it directly in your answer.

Answer

1. $\frac{\partial E^{(t)}}{\partial U}$. We are given the cross-entropy loss at time step t as:

$$E^{(t)} = -\sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$

where the predicted probability $\hat{y}^{(t)}$ is computed via a softmax over:

$$z^{(t)} = h^{(t)}U + b_2, \quad \hat{y}_j^{(t)} = \frac{\exp(z_j^{(t)})}{\sum_{k=1}^{|V|} \exp(z_k^{(t)})}$$

Thus:

$$\begin{aligned}\frac{\partial E^{(t)}}{\partial U} &= \frac{\partial z^{(t)}}{\partial U} \cdot \frac{\partial E^{(t)}}{\partial z^{(t)}} \\ &= \frac{\partial (h^{(t)}U + b_2)}{\partial U} \cdot (\hat{y}^{(t)} - y^{(t)}) \\ &= (h^{(t)})^\top \cdot (\hat{y}^{(t)} - y^{(t)})\end{aligned}$$

2. $\frac{\partial E^{(t)}}{\partial b_2}$. We are given:

$$E^{(t)} = -\sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}, \quad \text{where} \quad \hat{y}^{(t)} = \text{softmax}(z^{(t)}), \quad z^{(t)} = h^{(t)}U + b_2$$

As before, we apply the standard result:

$$\frac{\partial E^{(t)}}{\partial z^{(t)}} = \hat{y}^{(t)} - y^{(t)}$$

Since $z^{(t)} = h^{(t)}U + b_2$, and b_2 is added element-wise:

$$\frac{\partial z^{(t)}}{\partial b_2} = \mathbf{1}$$

Thus, by the chain rule:

$$\begin{aligned}\frac{\partial E^{(t)}}{\partial b_2} &= \frac{\partial E^{(t)}}{\partial z^{(t)}} \cdot \frac{\partial z^{(t)}}{\partial b_2} \\ &= \frac{\partial E^{(t)}}{\partial z^{(t)}} \cdot 1 \\ &= \hat{y}^{(t)} - y^{(t)} \\ &= \sum_{j=1}^{|V|} y_j^{(t)} - \hat{y}_j^{(t)}\end{aligned}$$

3. $\frac{\partial E^{(t)}}{\partial I}\big|_{(t)}$

$$\begin{aligned}\frac{\partial E^{(t)}}{\partial I} &= \frac{\partial h^{(t)}}{\partial I} \cdot \frac{\partial E^{(t)}}{\partial h^{(t)}} \\ &= \left[\sigma'(a^{(t)}) \cdot \frac{\partial(e^{(t)}I)}{\partial I}\right] \cdot (\hat{y}^{(t)} - y^{(t)})U^\top \\ &= \left[h^{(t)} \cdot (1 - h^{(t)})\right] \cdot (e^{(t)})^\top \cdot (\hat{y}^{(t)} - y^{(t)})U^\top \\ &= (e^{(t)})^\top \cdot \left[(\hat{y}^{(t)} - y^{(t)})U^\top \odot h^{(t)} \cdot (1 - h^{(t)})\right]\end{aligned}$$

4. $\frac{\partial E^{(t)}}{\partial H}\big|_{(t)}$

$$\begin{aligned}\frac{\partial E^{(t)}}{\partial H}\bigg|_{(t)} &= \frac{\partial E^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial a^{(t)}} \cdot \frac{\partial a^{(t)}}{\partial H} \\ &= (\hat{y}^{(t)} - y^{(t)})U^\top \odot \left(h^{(t)} \cdot (1 - h^{(t)})\right) \cdot (h^{(t-1)}) \\ &= (h^{(t-1)})^\top \cdot \left[(\hat{y}^{(t)} - y^{(t)})U^\top \odot h^{(t)} \cdot (1 - h^{(t)})\right]\end{aligned}$$

5. $\frac{\partial E^{(t)}}{\partial b_1}\big|_{(t)}$

$$\begin{aligned}\frac{\partial E^{(t)}}{\partial b_1}\bigg|_{(t)} &= \frac{\partial E^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial a^{(t)}} \cdot \frac{\partial a^{(t)}}{\partial b_1} \\ &= \frac{\partial E^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial \sigma(h^{(t-1)}H + e^{(t)}I + b_1)}{\partial(h^{(t-1)}H + e^{(t)}I + b_1)} \cdot \frac{\partial(h^{(t-1)}H + e^{(t)}I + b_1)}{\partial b_1} \\ &= (\hat{y}^{(t)} - y^{(t)})U^\top \odot \left(h^{(t)} \cdot (1 - h^{(t)})\right) \cdot 1 \\ &= (\hat{y}^{(t)} - y^{(t)})U^\top \odot \left(h^{(t)} \cdot (1 - h^{(t)})\right)\end{aligned}$$

6. $\frac{\partial E^{(t)}}{\partial h^{(t-1)}}$

$$\begin{aligned}\frac{\partial E^{(t)}}{\partial h^{(t-1)}} &= \frac{\partial E^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial a^{(t)}} \cdot \frac{\partial a^{(t)}}{\partial h^{(t-1)}} \\ &= (\hat{y}^{(t)} - y^{(t)})U^\top \odot \left(h^{(t)} \cdot (1 - h^{(t)})\right) \cdot H^\top\end{aligned}$$

(c) (10 points) To evaluate a language model, we use *perplexity*, which is defined as the inverse probability of the target word according to the model prediction $\bar{P}$:

$$PP^{(t)}(y^{(T)}, \hat{y}^{(t)}) = \frac{1}{\bar{P}(x_{t+1}^{pred} = x_{t+1}|x_t, \ldots, x_1)} = \frac{1}{\prod_{j=1}^{|V|}(\hat{y}_j^{(t)})^{y_j^{(t)}}}.$$

Show the relationship between *cross-entropy* and *perplexity.*

Answer. Cross-entropy at time step t is defined as:

$$CE^{(t)} = -\sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$

where $y^{(t)}$ is the one-hot encoding of the true next word, and $\hat{y}^{(t)}$ is the predicted probability distribution over the vocabulary. Since $y^{(t)}$ is one-hot, this simplifies to:

$$CE^{(t)} = -\log \bar{P}(x_{t+1} \mid x_1, \ldots, x_t) = -\log \hat{y}_{j^*}^{(t)}, \quad \text{where } j^* \text{ is the index of the correct word.}$$

Perplexity at time step t is defined as the inverse probability assigned to the correct word by the model:

$$PP^{(t)} = \frac{1}{\prod_{j=1}^{|V|}(\hat{y}_j^{(t)})^{y_j^{(t)}}}$$

This is equivalent to:

$$PP^{(t)} = \exp\left(-\sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}\right) = \exp\left(CE^{(t)}\right)$$

Therefore, the relationship between cross-entropy and perplexity is:

$$PP^{(t)} = \exp\left(CE^{(t)}\right) \quad \Longleftrightarrow \quad CE^{(t)} = \log\left(PP^{(t)}\right)$$

And for a sequence of length T, the average cross-entropy and overall perplexity relate as:

$$\text{Perplexity} = \exp\left(\frac{1}{T}\sum_{t=1}^{T} CE^{(t)}\right)$$

2. Option B (40 points) For this task, you will supervise-finetune a LLM (e.g., LLaMA, OPT, etc.) on a downstream task. You will use TinyLlama 1.1B model, which is a smaller version of LLaMA. Here is a tutorial on how to finetune LLaMA on a downstream task: `https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/TinyLlama_(1.1B)-Alpaca.ipynb"`. The downstream task is entity recognition (NER). The task is to extract Disease entities from a given sentence. The details for this assignment are included in this kaggle challenge: `https://www.kaggle.com/competitions/coms-6730-sft-llm-challenge`. You can use the provided dataset to finetune the model. The datasets are in the form of JSON files.

   The score distribution of the challenge is as follows:

   - 10 points: Convert dataset into the format required by supervise-finetuning LLM.

- 10 points: Finetune the model on the dataset and save the model.
- 20 points: Evaluate the model on the test set and submit the results to the kaggle challenge.
- 10 points (extra): Successful submission with more than 0.3 F1 score
- 10 points (extra): Successful submission with more than 0.5 F1 score

Please submit your notebook to canvas. The notebook should include every step of your work, including the code to convert the dataset, finetune the model, and evaluate the model. You can use any libraries you want, but make sure to include all the dependencies in your notebook.