ECE-GY 9123 / Spring 2021

Homework 4

Please upload your assignments on or before April 2, 2021.

- You are encouraged to discuss ideas with each other; but
- you **must acknowledge** your collaborator, and
- you **must compose your own** writeup and/or code independently.
- We **require** answers to theory questions to be written in LaTeX, and answers to coding questions in Python (Jupyter notebooks)
- Upload your answers in the form of a single PDF on Gradescope.

---

1. **(3 points)** *RNNs for images*. In this exercise, we will develop an RNN-type architecture for processing multi-dimensional data (such as RGB images). Here, hidden units are themselves arranged in the form of a grid (as opposed to a sequence). Each hidden unit is connected from the corresponding node from the input layer, as well as recurrently connected to its "top" and "left" neighbors. Here is a picture:

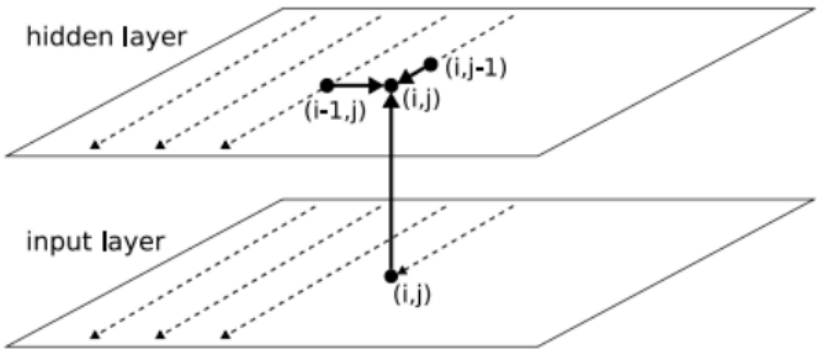

Figure 1: 2D RNNs

The equation for the hidden unit is given as follows (assume no bias parameters for simplicity):

$$h^{i,j} = \sigma\left(W_{in}x^{i,j} + W_{left}h^{i-1,j} + W_{top}h^{i,j-1}\right).$$

   a. Assume that the input grid is $n \times n$, each input $x^{i,j}$ is a c-channel vector, and the hidden state $h^{i,j}$ has dimension h. How many trainable parameters does this architecture have? You can assume zero padding as needed to avoid boundary effects.

   b. How many arithmetic operations (scalar adds and multiplies) are required to compute all the hidden activations? You can write your answer in big-Oh notation.

   c. Compare the above architecture with a regular convnet, and explain advantages/disadvantages.

2. **(1 points)** *Attention! My code takes too long.* In class, we showed that a regular self-attention layer takes $O(T^2)$ time to evaluate for an input with T tokens. Propose a way to reduce this running time to, say, $O(T)$, and comment on its possible pros vs cons.

3. **(2 points)** *Making skip-gram training practical*. Suppose we have a dictionary containing d words, and we are trying to learn skip-gram embeddings of each word using a very large training corpus. Suppose that our embedding dimension is chosen to be h.

    a. Calculate the gradient of the cross-entropy loss for a *single* target-context training pair of words. What is the running time of calculating this gradient in terms of d and h?

    b. Here is a second approach. Argue that the gradient from part a can be written as a weighted average of some number of terms, and intuitively describe a *sampling*-based method to improve the running time of the gradient calculation.

4. **(4 points)** Open the (incomplete) Jupyter notebook provided as an attachment to this homework in Google Colab (or other cloud service of your choice) and complete the missing items. Save your finished notebook in PDF format and upload along with your answers to the above theory questions in a single PDF.