CS-GY 9223G / Fall 2020

Homework 4

1. **(3 points)** *Pay attention.* Consider the standard (scaled) dot-product self-attention mechanism that computes alignment scores between all pairs of input symbols; so if there are n tokens in a sequence this require computing n^2 query-key dot products. In this problem, let us try to make this more efficient.

   a. Consider *autoregressive* self-attention where every token only attends to its own position and all previous positions. Calculate how many dot-products are now required as a function of n. Draw an $n \times n$ grid and shade/mark the squares that depict the calculated attention scores.

   b. Consider *strided* self-attention where every token attends to *at most* t positions prior to it, plus itself. Calculate how many dot-products are required as a function of n and t. Draw a similar grid as above and mark the scores that are calculated.

   c. Consider *windowed* self-attention where the n tokens are partitioned into windows of size w (assume w divides n), and every token attends to all positions within its window and prior to it, plus itself. Draw a similar grid as above and mark the scores that are calculated.

   Solution

   a. $\sum_{i=1}^{n} = n(n+1)/2$.

   b. $\sum_{i=n-t}^{n} i = \frac{(2n-t)(t+1)}{2}$.

   c. $\frac{n}{w} \cdot \frac{w(w+1)}{2} = \frac{n(w+1)}{2}$.

   The figures are as follows (taken from "Generating Long Sequences with Sparse Transformers" by Child et al, 2019.)

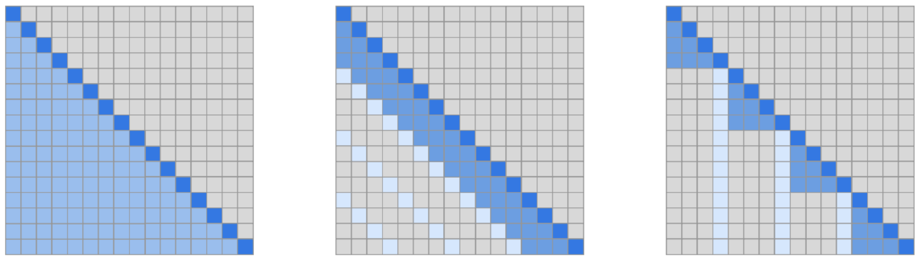

Figure 1: Attention patterns ($t = 3, w = 4$)

2. **(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:

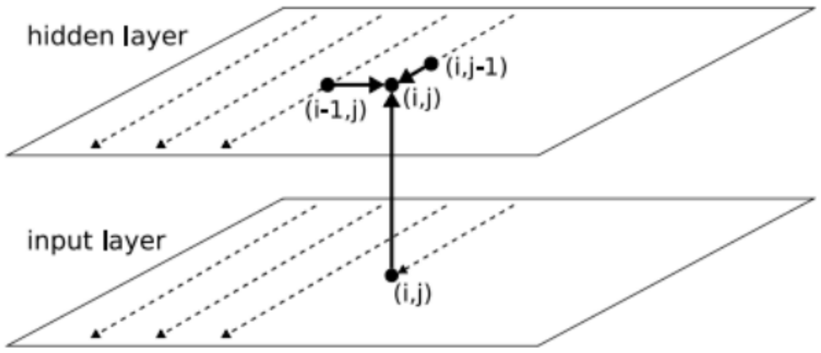

Figure 2: 2D RNNs

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?

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.

Solution

a. $2h^2 + hc$ (the weight matrix mapping input to hidden state is a $c \times h$ matrix, and the weight matrix mapping hidden states is $h \times h$).

b. Each hidden state is defined in terms of 3 matrix-vector products, hence takes $O(h^2 + hc)$ scalar adds and multiplies to compute. Therefore, the overall number of trainable parameters is $O(h^2n^2 + hcn^2)$.

c. Pros of 2D-RNN over convnets: Directionality (so extracted features are naturally time sensitive; this could be a pro or con depending on how you look at it). Outputs are context-sensitive. Suitable for tasks such as image completion.

Cons: Not easily parallelizable. Harder to train (due to unrolling).

3. **(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.

Solution

A solution notebook has been added to NYUClasses.