CS4803/7643: Deep Learning
Fall 2018
Problem Set 3

Instructor: Dhruv Batra
TAs: Nirbhay Modhe, Eric Wijmans, Harsh Agrawal, Michael Cogswell
Discussions: `http://piazza.com/gatech/fall2018/cs48037643`

Due: 11:55pm on Tuesday, November 6, 2018

Instructions

1. We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!
   - Each problem/sub-problem should be on at least one or more pages i.e. start evert problem/sub-problem on a fresh page.
   - When submitting to Gradescope, make sure to mark which page corresponds to each problem/sub-problem.
   - For the coding problem, please use the provided `collect_submission.sh` script and upload `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), 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 Convolution Basics

The convolution layer inside of a CNN is intrinsically an *affine transformation*: A vector is received as input and is multiplied with a matrix to produce an output (to which a bias vector is usually added before passing the result through a nonlinearity). This operation can be represented as $y = Ax$, in which A describes the affine transformation.

1. **[2 points]** We will first revisit the convolution layer as discussed in the class. Consider a convolution layer with a 2x2 kernel W, operated on a single input channel X, represented as:

$$W = \begin{bmatrix} w_{(0,0)}, w_{(0,1)} \\ w_{(1,0)}, w_{(1,1)} \end{bmatrix}, X = \begin{bmatrix} x_{(0,0)}, x_{(0,1)}, x_{(0,2)} \\ x_{(1,0)}, x_{(1,1)}, x_{(1,2)} \\ x_{(2,0)}, x_{(2,1)}, x_{(2,2)} \end{bmatrix} \tag{1}$$

   Now let us work out a **stride-3** convolution layer, with **zero padding size of 1**. Consider 'flattening' the input tensor X in row-major order as:

$$X = \left[x_{(0,0)}, x_{(0,1)}, x_{(0,2)}, ..., x_{(2,0)}, x_{(2,1)}, x_{(2,2)}\right]^\top \tag{2}$$

   Write down the convolution as a matrix operation A that: $Y = AX$. Output Y is also flattened in row-major order.

2. **[2 points]** Recall that transposed convolution can help us upsample the feature size spatially. Consider a transposed convolution layer with a 2x2 kernel W operated on a single input channel X, represented as:

$$W = \begin{bmatrix} w_{(0,0)}, w_{(0,1)} \\ w_{(1,0)}, w_{(1,1)} \end{bmatrix}, X = \begin{bmatrix} x_{(0,0)}, x_{(0,1)} \\ x_{(1,0)}, x_{(1,1)} \end{bmatrix} \tag{3}$$

   We 'flattened' the input tensor in row-major order as $X = [x_{(0,0)}, x_{(0,1)}, x_{(1,0)}, x_{(1,1)}]$.

   Write down the affine transformation A corresponding to a transposed convolution layer with kernel W, **stride 2, no padding**. Output Y is also flattened in row-major order.

3. **[4 points]** Convolution layers in most CNNs consist of multiple input and output feature maps. The collection of kernels form a 4D tensor (output channels o, input channels i, filter rows k, filter columns k), represented in short as (o, i, k, k). For each output channel, each input channel is convolved with a distinct 2D slice of the 4D kernel and the resulting set of feature maps is summed element-wise to produce the corresponding output feature map.

   There is an interesting property that a convolutional layer with kernel size $(o \times r^2, i, k, k)$ is identical to a transposed convolution layer with kernel size $(o, i, k \times r, k \times r)$. Here the word 'identical' means with the same input feature X, both operations will give the same output Y with only a difference in the ordering of flattened elements of Y.

   Now let us prove the property in a restricted setting. Consider $o = 1, r = 2, i = 1, k = 1$. Given the same input feature X as in equation 3, write down the affine transformation for a convolutional layer with kernel size (4, 1, 1, 1), and show it is an operation identical to a transposed convolution layer with kernel size $(1, 1, 2, 2)$.

2 Directed Acyclic Graphs (DAG) [Extra credit for 4803 and 7643]

One important property for feed-forward network that we have discussed in class is that it must be a directed acyclic graph (DAG). Recall that a *DAG is a directed graph that contains no directed cycles.* We will study some of its properties in this question.
Define $G = (V, E)$ in which V is the set of all nodes as $\{v_1, v_2, ..., v_i, ...v_n\}$ and E is the set of edges $E = \big\{e_{i,j} = (v_i, v_j) \mid v_i, v_j \in V\big\}$.
A *topological order of a directed graph* $G = (V, E)$ is an ordering of its nodes as $\{v_1, v_2, ..., v_i, ...v_n\}$ so that for every edge (v_i, v_j) we have $i < j$.
There are several lemmas can be inferred from the definition of DAG. One lemma is: if G is a DAG, then G has a node with no incoming edges.
Given the above lemma, prove the following two lemmas:

1. **[2 points]** If the graph G is a DAG, then G has a topological ordering.
2. **[2 points]** If the graph G has a topological order, then G is a DAG.

3 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 odd it returns 1, and 0 if it is even. For example, given input sequence $[0, 1, 0, 1, 1, 0]$, the parity sequence is $[0, 1, 1, 0, 1, 1]$. 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. **[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:

   $$h_t = W^\top h_{t-1} \tag{4}$$

   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:

   $$\rho(W) = \max\{|\lambda_1|, ..., |\lambda_n|\} \tag{5}$$

   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 $T \gg 0$

4 Coding: Sequence models for image captioning [30 points for CS7643, 20 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/AY2019/cs7643_fall/hw3/