CSCI-GA 2572 Deep Learning Homework 2: Convolutional Neural Networks and Recurrent Neural Networks

Haozhen Bo, N19289664

Spring 2022

1 Theory (50pt)

1.1 Convolutional Neural Netoworks (20 pts)

(a) (2 pts) Given an input image of dimension 10×11, what will be output dimension after applying a convolution with 5×5 kernel, stride of 2, and no padding?
For the width dimension, the output size is $\frac{11-5+2*0}{2}+1=4$.
For the height dimension, the output size is $\frac{10-5+2*0}{2}+1=3.5$, which is not an integer. Hence, the input should be cropped and the output size will be 3.
Thus, the output dimension should be 3×4.

(b) (3 pts) Given an input of dimension $C \times H \times W$, what will be the dimension of the output of a convolutional layer with kernel of size $K \times K$, padding P, stride S, dilation D, and F filters. Assume that $H \geq K, W \geq K$.
For the Channel dimension, the output will have size F, equal to the number of filters.
For the Height dimension, the outpu will have size $\lfloor \frac{H-[D\times(K-1)+1]+2P}{S}+1 \rfloor$.
For the Width dimension, the outpu will have size $\lfloor \frac{W-[D\times(K-1)+1]+2P}{S}+1 \rfloor$.

(c) (15 pts) In this section, we are going to work with 1-dimensional convolutions. Discrete convolution of 1-dimensional input $x[n]$ and kernel $k[n]$ is defined as follows:

$$s[n] = (x * k)[n] = \sum_m x[n-m]k[m]$$

However, in machine learning convolution is usually implemented as cross-correlation, which is defined as follows:

$$s[n] = (x * k)[n] = \sum_m x[n+m]k[m]$$

Note the difference in signs, which will get the network to learn an "flipped" kernel. In general it doesn't change much, but it's important to keep it in mind. In convolutional neural networks, the kernel $k[n]$ is usually 0 everywhere, except a few values near 0: $\forall_{|n|>M} k[n] = 0$. Then, the formula becomes:

$$s[n] = (x * k)[n] = \sum_{m=-M}^{M} x[n+m]k[m]$$

Let's consider an input $x[n]$, $x : \{1,2,3,4,5,6,7\} \to \mathbb{R}^2$ of dimension 7, with 2 channels, and a convolutional layer f_W with one filter, with kernel size 5, stride of 2, no dilation, and no padding. The only parameters of the convolutional layer is the weight W, $W \in \mathbb{R}^{1\times 2\times 5}$, there's no bias and no nonlinearity.

(i) (2 pts) What is the dimension of the output $f_W(x)$? Provide an expression for the value of elements of the convolutional layer output $f_W(x)$. Example answer format here and in the following sub-problems: $f_W(x) \in \mathbb{R}^{42\times 42\times 42}$, $f_W(x)[i,j,k] = 42$.

$f_W(x) \in \mathbb{R}^{2\times 1}, x \in \mathbb{R}^{7\times 2}, W \in \mathbb{R}^{5\times 2}$.
$f_W(x)[i,1] = \sum_{j=1}^{5}\sum_{k=1}^{2} x[j+2i-2,k]W[j,k], j = 1,2.$

(ii) (4 pts) What is the dimension of $\frac{\partial f_W(x)}{\partial W}$? Provide an expression for the values of the derivative $\frac{\partial f_W(x)}{\partial W}$.
Since $f_W(x) \in \mathbb{R}^{2\times 1}, W \in \mathbb{R}^{5\times 2}$. Then, $\frac{\partial f_W(x)}{\partial W}$ should be $\mathbb{R}^{2\times 5\times 2}$.
$\frac{\partial f_W(x)}{\partial W}[i,j,k] = x[j+2i-2,k], i = 1,2, j = 1,...,5, k = 1,2.$

(iii) (4 pts) What is the dimension of $\frac{\partial f_W(x)}{\partial x}$? Provide an expression for the values of the derivative $\frac{\partial f_W(x)}{\partial x}$.
Since $f_W(x) \in \mathbb{R}^{2\times 1}, x \in \mathbb{R}^{7\times 2}$. Then, $\frac{\partial f_W(x)}{\partial x}$ should be $\mathbb{R}^{2\times 7\times 2}$.
$$\frac{\partial f_W(x)}{\partial x}[i,j,k] = \begin{cases} W[j,k] & ,i = 1,2, j = 2i-1,...,2i+3 \\ 0 & ,otherwise. \end{cases} , k = 1,2.$$

(iv) (5 pts) Now, suppose you are given the gradient of the loss ℓ w.r.t. the output of the convolutional layer $f_W(x)$, *i.e.* $\frac{\partial \ell}{\partial f_W(x)}$. What is the dimension of $\frac{\partial \ell}{\partial W}$? Provide an expression for $\frac{\partial \ell}{\partial W}$. Explain similarities and differences of this expression and expression in (i).
Since $W \in \mathbb{R}^{5\times 2}$, if we follow the denominator layout, then $\frac{\partial \ell}{\partial W}$ should be $\mathbb{R}^{5\times 2}$.

$$\begin{aligned}\frac{\partial \ell}{\partial W}[j,k] &= \sum_{i=1,2} \frac{\partial \ell}{\partial f_W(x)}[i] \frac{\partial f_W(x)}{\partial W}[i,j,k] \\ &= \sum_{i=1,2} \frac{\partial \ell}{\partial f_W(x)}[i] x[j+2i-2,k]\end{aligned}$$

This shows that $\frac{\partial \ell}{\partial W}$ is the convolution of input x with kernel $\frac{\partial \ell}{\partial f_W(x)}$, but in stride of 1, dilation of 2.
In part (i), it's the convolution of input x with kernel W, in stride of 2, dilation of 0.

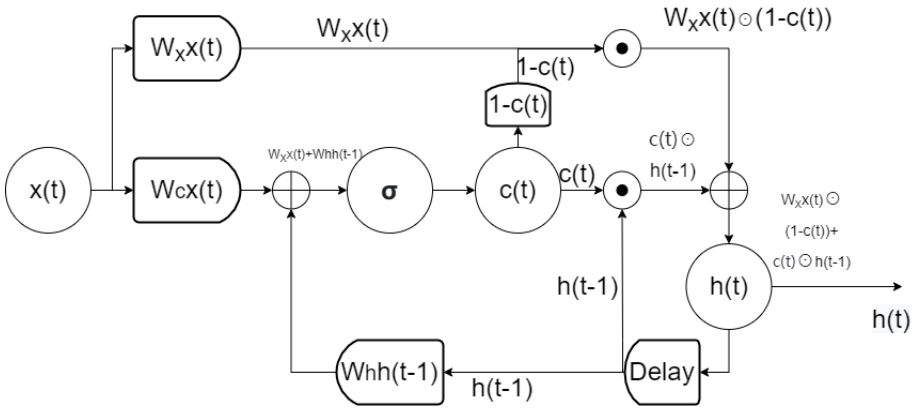
1.2 Recurrent Neural Networks (20 pts)

In this section we consider a simple recurrent neural network defined as follows:

$$c[t] = \sigma(W_c x[t] + W_h h[t-1]) \tag{1}$$
$$h[t] = c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t] \tag{2}$$

where σ is element-wise sigmoid, $x[t] \in \mathbb{R}^n$, $h[t] \in \mathbb{R}^m$, $W_c \in \mathbb{R}^{m\times n}$, $W_h \in \mathbb{R}^{m\times m}$, $W_x \in \mathbb{R}^{m\times n}$, $\odot$ is Hadamard product, $h[0] \doteq 0$.

(a) (5 pts) Draw a diagram for this recurrent neural network, similar to the diagram of RNN we had in class. We suggest using diagrams.net.



(b) (2pts) What is the dimension of $c[t]$?
$c \in \mathbb{R}^m$.

(c) (10 pts) Suppose that we run the RNN to get a sequence of $h[t]$ for t from 1 to K. Assuming we know the derivative $\frac{\partial \ell}{\partial h[t]}$, provide dimension of and an expression for values of $\frac{\partial \ell}{\partial W_x}$. What are the similarities of backward pass and forward pass in this RNN?
$\frac{\partial \ell}{\partial W_x}$ should have size $m \times n$, the same as W_x in denominator layout.

$$\begin{aligned}
\frac{\partial \ell}{\partial W_x} &= \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial \ell}{\partial h(t)} \frac{\partial h(t)}{\partial h(k)} \frac{\partial h(k)}{\partial W_x} \\
&= \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial \ell}{\partial h(t)} \prod_{j=k}^{t-1} \frac{\partial h(j+1)}{\partial h(j)} \frac{\partial h(k)}{\partial W_x} \\
&= \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial \ell}{\partial h(t)} \prod_{j=k}^{t-1} \left\{[h(j) - W_x x(j+1)] \odot W_h^T \sigma'(W_c x(j+1) + W_h h(j)) + c(j+1)\right\} \frac{\partial W_x x(k)}{\partial W_x} \odot [1 - c(k)]
\end{aligned}$$

Both the forward and backward passes are recurrent. Past information is used to calculate the output at time T and the derivative of error to parameter. Also, assume $\sigma(c(t)) = 0$ or 1 in each iteration $t = T$. Then in the forward pass, the output is either $W_x x(T)$ or $h(T-1)$; in the backward pass, the backpropagation trace back along only one path, which is the exact path of the forward pass just in the opposite direction.

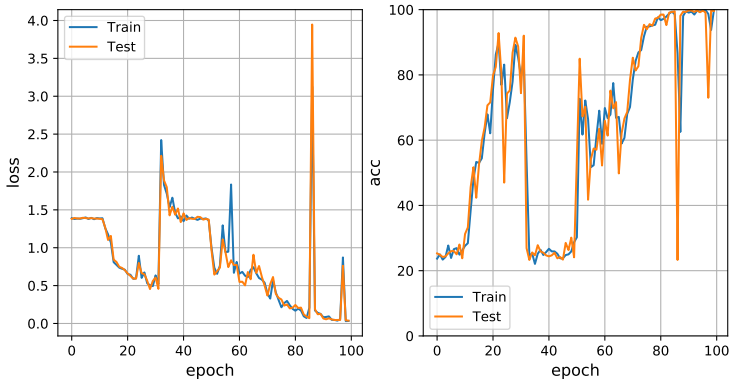
(d) (3pts) Can this network be subject to vanishing or exploding gradients? Why?
The reason for vanishing gradient in general RNN comes from the $\prod_{j=k}^{t-1} \frac{\partial h(j+1)}{\partial h(j)}$ part. Multiplying a matrix for multiple times may cause an exponential decreasing of the value. However, this network shall be less subject to vanishing gradients than the general RNN. See the part of production in the gradient expression:

$$\prod_{j=k}^{t-1} \left\{[h(j) - W_x x(j+1)] \odot W_h^T \sigma'(W_c x(j+1) + W_h h(j)) + c(j+1)\right\}.$$

This is an addition of some matrix and $c(j+1)$, which is less likely to be exponentially decreasing, as there is some parameters that the network can learn such that the gradient will not vanish.
However, this network shall be subject to exploding gradients. The gate structure does not prevent exploding.

1.3 Debugging loss curves (10pts)

In the class on Wednesday, February 16, when working with notebook `08-seq_classification`, we saw RNN training curves. In Section 8 "Visualize LSTM", we observed some "kinks" in the loss curve.



1. (2pts) What caused the spikes on the left?
   During SGD, there happens to be an abruptly change in the gradient. So the gradient calculated may point to the opposite direction of optimizing, and the magnitude may be large. So, after taking the step, the parameters are even worse.

2. (2pts) How can they be higher than the initial value of the loss?
   The initial loss is calculated by a random initial parameters. Here, it is possible that after taking the bad step, we get a set of worse parameters.

3. (3pts) What are some ways to fix them?
   By changing learning rate, batch size, clipping the gradient or using momentum.

4. (3pts) Explain why the loss and accuracy are at these values before training starts. You may need to check the task definition in the notebook.
   The task is classification with 4 classes. So, the probability success of random guess is 0.25, which is the initial accuracy. The loss function is cross entropy $L = -\sum y \log(\hat{y}) = -\log(\frac{1}{4}) = \log(4) \approx 1.39$.

2 Implementation (50pts + 5pts extra credit)

There are three notebooks in the practical part:

- (25pts) Convolutional Neural Networks notebook: `hw2_cnn.ipynb`
- (20pts) Recurrent Neural Networks notebook: `hw2_rnn.ipynb`
- (5pts + 5pts extra credit) : This builds on Section 1.3 of the theoretical part.
  - (5pts) Change the model training procedure of Section 8 in `08-seq_classification` to make the training curves have no spikes. You should only change the training of the model, and not the model itself or the random seed.
  - (5pts extra credit) Visualize the gradients and weights throughout training before and after you fix the training procedure.

Plase use your NYU Google Drive account to access the notebooks. First two notebooks contain parts marked as `TODO`, where you should put your code. These notebooks are Google Colab notebooks, you should copy them to your drive, add your solutions, and then download and submit them to NYU Brightspace. The notebook from the class, if needed, can be uploaded to Colab as well.