Homework 2: Convolutional Neural Networks and Recurrent Neural Networks

CSCI-GA 2572 Deep Learning

Spring 2021

The goal of homework 2 is to get you to work with convolutional neural networks and recurrent neural networks.

In the theoretical part, you will work on figuring out how backpropagation works in these networks. In part 2, we will implement and train them.

In part 1, you should submit all your answers in a pdf file. As before, we recommend using LaTeX.

For part 2, you will implement some neural networks by adding your code to the provided ipynb file.

As before, please use numerator layout.

The due date of homework 2 is 11:55pm `03/11`. Submit the following files in a zip file `your_net_id.zip` through NYU classes:

- `hw2_theory.pdf`
- `hw2_cnn.ipynb`
- `hw2_rnn.ipynb`

The following behaviors will result in penalty of your final score:

1. 10% penalty for submitting your file without using the correct naming format (including naming the zip file, PDF file or python file wrong, adding extra files in the zip folder, like the testing scripts in your zip file).
2. 20% penalty for late submission within the first 24 hours after the deadline. We will not accept any late submission after the first 24 hours.
3. 20% penalty for code submission that cannot be executed following the steps we mentioned.

1 Theory (50pt)

1.1 Convolutional Neural Netoworks (30 pts)

(a) (5 pts) Given an input image of dimension 10×11, what will be output dimension after applying a convolution with 3×3 kernel, stride of 2, and no padding?

Answer: 4×5

(b) (5 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 $H \geq K$, $W \geq K$.

Answer:

$$F \times \left(\left\lfloor \frac{(H + 2P - (D(K-1)+1)}{S} \right\rfloor + 1 \right) \times \left(\left\lfloor \frac{(W + 2P - (D(K-1)+1)}{S} \right\rfloor + 1 \right)$$

(c) (20 pts) For 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 usually is implemented as a 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 $x[n]$, $x : \{1,2,3,4,5\} \to \mathbb{R}^2$ of dimension 5, with 2 channels, and a convolutional layer f_W with one filter, with kernel size 3, stride of 2, no dilation, and no padding. The only parameters of the convolutional layer is the weight W, $W \in \mathbb{R}^{1\times 2\times 3}$, there's no bias and no non-linearity.

(i) (4 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: $f_W(x) \in \mathbb{R}^{42\times 42\times 42}$, $f_W(x)[i,j,k] = 42$.

Answer:

$$f_W(x) \in \mathbb{R}^2$$

$$f_W(x)[r] = \sum_{k=1}^{2} \sum_{i=1}^{3} x[2(r-1)+i][k] W_{k,i}$$

(ii) (4 pts) What is the dimension of $\frac{\partial f_W(x)}{\partial W}$? Provide an expression for the values of the derivative $\frac{\partial f_W(x)}{\partial W}$.

Answer: Here, because in numerator layout we'd need to transpose W, but it's unclear how that's done if W is a tensor, multiple possible solutions exist. Below is one where we don't transpose it's dimension at all, but solutions where transpose is done should be similar except for the order of indices.

$$\frac{\partial f_W(x)}{\partial W} \in \mathbb{R}^{(2)\times(1\times 2\times 3)}$$

$$\frac{\partial f_W(x)}{\partial W}[r,c,k,i] = x[2(r-1)+i][k]$$

(iii) (6 pts) What is the dimension of $\frac{\partial f_W(x)}{\partial x}$? Provide an expression for the values of the derivative $\frac{\partial f_W(x)}{\partial x}$.

Answer:

$$\frac{\partial f_W(x)}{\partial x} \in \mathbb{R}^{(2)\times(2\times 5)}$$

$$\frac{\partial f_W(x)}{\partial x}[r,k,i] = \begin{cases} W_{1,k,i-2(r-1)} \text{ if } 1 \le i-2(r-1) \le 3 \\ 0 \text{ otherwise} \end{cases}$$

(iv) (6 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).

Answer:

$$\frac{\partial \ell}{\partial W} \in \mathbb{R}^{1\times 2\times 3}$$

$$\left(\frac{\partial \ell}{\partial W}\right)[1,k,i] = \sum_{r=1}^{2} \left(\frac{\partial \ell}{\partial f_W(x)}\right)[r] x[2(r-1)+i,k]$$

This is a dilated convolution. Both forward and backward pass of a convolutional layer consist in applying a convolution. The difference is that stride becomes a dilation when peforming backward pass.

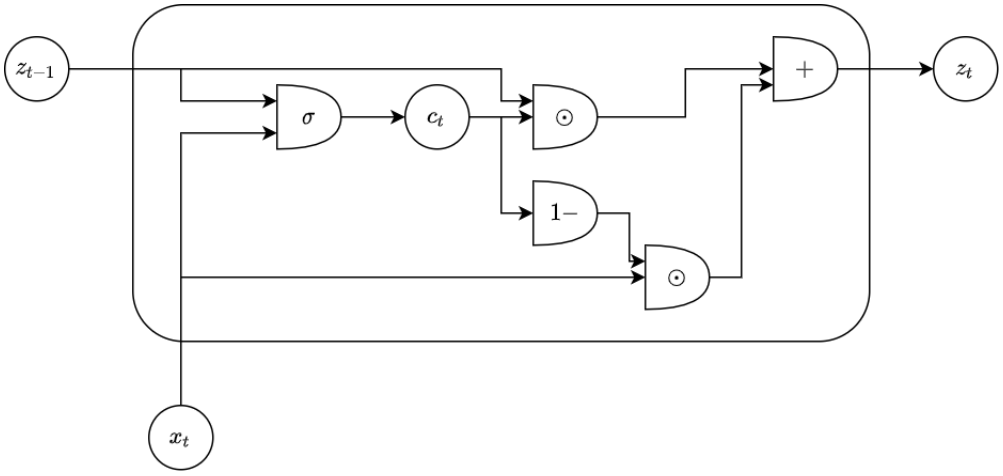
1.2 Recurrent Neural Networks (20 pts)

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

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

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

(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.



Answer:

(b) (2pts) What is the dimension of $c[t]$?

Answer: $c_t \in \mathbb{R}^m$

(c) (10 pts) Suppose that we run the RNN to get a sequence of $z[t]$ for t from 1 to K. Assuming we know the derivative $\frac{\partial \ell}{\partial z[t]}$, 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?

Answer:

$$\frac{\partial \ell}{\partial W_x} \in \mathbb{R}^{n \times m}$$

$$\frac{\partial \ell}{\partial W_x} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial z[t]} \left(\overline{\frac{\partial z[t]}{\partial W_x}} + \sum_{i=1}^{t-1} \left(\prod_{j=i}^{t-1} \frac{\partial z[i+1]}{\partial z[i]} \right) \overline{\frac{\partial z[i]}{\partial W_x}} \right)$$

$$\overline{\frac{\partial z[t]}{\partial W_x}} = (1 - c[t]) \odot P$$

$$\frac{\partial z[t]}{\partial z[t-1]} = \text{diag}(c[t]) + \text{diag}(z[t-1]) \frac{\partial c[t]}{\partial z[t-1]} - \text{diag}(W_x x[t]) \frac{\partial c[t]}{\partial z[t-1]}$$

$$\frac{\partial c[t]}{\partial z[t-1]} = \text{diag}\left(\sigma(W_c x[t] + W_z z[t-1]) \odot (1 - \sigma(W_c x[t] + W_z z[t-1]))\right) W_z^\top$$

Where P is a tensor, $P \in \mathbb{R}^{m \times n \times m}$,

$$P_i = \begin{bmatrix} 0 & \times & i-1 & x[t] & \dots & 0 & \dots \end{bmatrix}$$

In other words P_i is a zero matrix except i-th column, which is equal to $x[t]$. $\overline{\frac{\partial z[t]}{\partial W_x}}$ means partial derivative with $z[t-1]$ fixed to a constant. Forward pass and backward pass are similar because they both contain computing values recurrently.

(d) (3pts) Can this network be subject to vanishing or exploding gradients? Why?

Answer: This RNN cannot be subject to exploding gradients because the state vector z_t doesn't get multiplied by any matrix from one timestep to the next. This RNN can be subject to vanishing gradients however because on each iteration the state vector z_t is element-wise multiplied with c_t, which is a vector of values between 0 and 1.

2 Implementation (50pt)

There are two notebooks in the practical part:

- Convolutional Neural Networks notebook: `hw2_cnn.ipynb`
- Recurrent Neural Networks notebook: `hw2_rnn.ipynb`

Plase use your NYU account to access the notebooks. Both 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 Classes.