Homework 2: Convolutional Neural Networks and Recurrent Neural Networks

DS-GA 1008 Deep Learning

Spring 2024

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.

Submit the following files in a zip file `your_net_id.zip` through NYU Brightspace:

- `hw2_theory.pdf`
- `hw2_moe.ipynb`
- `hw2_cnn.ipynb`
- `hw2_rnn.ipynb`
- `08-seq_classification.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 code submission that cannot be executed following the steps we mentioned.

1 Theory (50pt)

1.1 Convolutional Neural Netoworks (15 pts)

(a) (1 pts) Given an input image of dimension 11×21, what will be output dimension after applying a convolution with 5×4 kernel, stride of 4, and no padding?

5×2

(b) (2 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 $H \geq K$, $W \geq K$.

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

(c) (12 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 $x[n] \in \mathbb{R}^5$, with $1 \leq n \leq 7$, e.g. it is a length 7 sequence with 5 channels. We consider the 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, $W \in \mathbb{R}^{1 \times 5 \times 3}$, there's no bias and no non-linearity.

(i) (1 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: $f_W(x) \in \mathbb{R}^{42 \times 42 \times 42}$, $f_W(x)[i,j,k] = 42$.

$$f_W(x) \in \mathbb{R}^3$$

$$f_W(x)[r] = \sum_{k=1}^{5} \sum_{i=1}^{3} x[2(r-1)+i][k] W_{1,k,i}$$

(ii) (3 pts) What is the dimension of $\frac{\partial f_W(x)}{\partial W}$? Provide an expression for the values of the derivative $\frac{\partial f_W(x)}{\partial W}$.

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.

$$\frac{\partial f_W(x)}{\partial W} \in \mathbb{R}^{(3)\times(1\times5\times3)}$$

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

(iii) (3 pts) What is the dimension of $\frac{\partial f_W(x)}{\partial x}$? Provide an expression for the values of the derivative $\frac{\partial f_W(x)}{\partial x}$.

$$\frac{\partial f_W(x)}{\partial x} \in \mathbb{R}^{(3)\times(5\times7)}$$

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

(iv) (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).

$$\frac{\partial \ell}{\partial W} \in \mathbb{R}^{1\times5\times3}$$

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

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 (30 pts)

1.2.1 Part 1

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

diagram.png

(b) (1pts) What is the dimension of $c[t]$?

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

(c) (5 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?

$$\frac{\partial \ell}{\partial W_x} \in \mathbb{R}^{n \times m}$$
$$\frac{\partial \ell}{\partial W_x} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial h[t]} \left(\overline{\frac{\partial h[t]}{\partial W_x}} + \sum_{i=1}^{t-1} \left(\prod_{j=i}^{t-1} \frac{\partial h[i+1]}{\partial h[i]} \right) \overline{\frac{\partial h[i]}{\partial W_x}} \right)$$
$$\overline{\frac{\partial h[t]}{\partial W_x}} = (1 - c[t]) \odot P$$
$$\frac{\partial h[t]}{\partial h[t-1]} = \operatorname{diag}(c[t]) + \operatorname{diag}(h[t-1]) \frac{\partial c[t]}{\partial h[t-1]} - \operatorname{diag}(W_x x[t]) \frac{\partial c[t]}{\partial h[t-1]}$$
$$\frac{\partial c[t]}{\partial h[t-1]} = \operatorname{diag}\left(\sigma(W_c x[t] + W_h h[t-1]) \odot (1 - \sigma(W_c x[t] + W_h h[t-1]))\right) W_h^\top$$

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 $x[t]$. $\overline{\frac{\partial h[t]}{\partial W_x}}$ means partial derivative with $h[t-1]$ fixed to a constant. Forward pass and backward pass are similar because they both contain computing values recurrently.

(d) (2pts) Can this network be subject to vanishing or exploding gradients? Why?

This RNN cannot be subject to exploding gradients because the state vector h_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 h_t is element-wise multiplied with c_t, which is a vector of values between 0 and 1.

1.2.2 Part 2

We define an AttentionRNN(2) as

$$q_0[t], q_1[t], q_2[t] = Q_0 x[t], Q_1 h[t-1], Q_2 h[t-2] \tag{3}$$
$$k_0[t], k_1[t], k_2[t] = K_0 x[t], K_1 h[t-1], K_2 h[t-2] \tag{4}$$
$$v_0[t], v_1[t], v_2[t] = V_0 x[t], V_1 h[t-1], V_2 h[t-2] \tag{5}$$
$$w_i[t] = q_i[t]^\top k_i[t] \tag{6}$$
$$a[t] = \text{softargmax}([w_0[t], w_1[t], w_2[t]]) \tag{7}$$
$$h[t] = \sum_{i=0}^{2} a_i[t] v_i[t] \tag{8}$$

Where $x[t], h[t] \in \mathbb{R}^n$, and $Q_i, K_i, V_i \in \mathbb{R}^{n \times n}$. We define $h[t] = \mathbf{0}$ for $t < 1$. You may safely ignore these bases cases in the following questions.

(a) (4 pts) Draw a diagram for this recurrent neural network

diagram.png

(b) (1 pt) What is the dimension of $a[t]$? 3

(c) (3 pts) Extend this to, AttentionRNN(k), a network that uses the last k state vectors h. Write out the system of equations that defines it. You may use set notation or ellipses (...) in your definition.

$$q_0[t], q_1[t], \ldots, q_k[t] = Q_0 x[t], Q_1 h[t-1], \ldots, Q_k h[t-k] \tag{9}$$
$$k_0[t], k_1[t], \ldots, k_k[t] = K_0 x[t], K_1 h[t-1], \ldots, K_k h[t-k] \tag{10}$$
$$v_0[t], v_1[t], \ldots, v_k[t] = V_0 x[t], V_1 h[t-1], \ldots, V_k h[t-k] \tag{11}$$
$$w_i[t] = q_i[t]^\top k_i[t] \tag{12}$$
$$a[t] = \text{softargmax}([w_0[t], \ldots, w_k[t]]) \tag{13}$$
$$h[t] = \sum_{i=0}^{k} a_i[t] v_i[t] \tag{14}$$

(d) (3 pts) Modify the above network to produce AttentionRNN(∞), a network that uses every past state vector. Write out the system of equations that defines it. You may use set notation or ellipses (...) in your definition. HINT: We can do this by tying together some set of parameters, e.g. weight sharing.

$$q_0[t], q_1[t], \ldots, q_{t-1}[t] = Q_0 x[t], Qh[t-1], \ldots, Qh[1] \tag{15}$$
$$k_0[t], k_1[t], \ldots, k_{t-1}[t] = K_0 x[t], Kh[t-1], \ldots, Kh[1] \tag{16}$$
$$v_0[t], v_1[t], \ldots, v_{t-1}[t] = V_0 x[t], Vh[t-1], \ldots, Vh[1] \tag{17}$$
$$w_i[t] = q_i[t]^\top k_i[t] \tag{18}$$
$$a[t] = \text{softargmax}([w_0[t], \ldots, w_{t-1}[t]]) \tag{19}$$
$$h[t] = \sum_{i=0}^{t-1} a_i[t] v_i[t] \tag{20}$$

(e) (5 pts) Suppose the loss ℓ is computed. Please write down the expression for $\frac{\partial h[t]}{\partial h[t-1]}$ for AttentionRNN(2).

$\frac{\partial h[t]}{\partial h[t-1]}$ with $h[t] = \sum_{i=1}^{2} a_i[t] v_i[t]$ has

$h[t] = a_0[t]v_0[t] + a_1[t]v_1[t] + a_2[t]v_2[t]$.

From the definitions we have

$$a_0[t] = \frac{\exp(q_0[t]^\intercal k_0[t])}{\sum_{i=0}^{2} \exp(q_i[t]^\intercal k_i[t])}$$
$$a_1[t] = \frac{\exp(q_1[t]^\intercal k_1[t])}{\sum_{i=0}^{2} \exp(q_i[t]^\intercal k_i[t])}$$
$$a_2[t] = \frac{\exp(q_2[t]^\intercal k_2[t])}{\sum_{i=0}^{2} \exp(q_i[t]^\intercal k_i[t])}$$

and that

$$v_0[t] = V_0 x[t] \qquad v_1[t] = V_1 h[t-1] \qquad v_2[t] = V_2 h[t-2]$$

$\frac{\partial h[t]}{\partial h[t-1]}$ can be split into three expressions:

$$\frac{\partial (a_0[t]v_0[t])}{\partial h[t-1]} \qquad \frac{\partial (a_1[t]v_1[t])}{\partial h[t-1]} \qquad \frac{\partial (a_2[t]v_2[t])}{\partial h[t-1]}$$

Note that only q_1, k_1, v_1 are depend on $h[t-1]$. Therefore, the first and third expressions depend only on the denominator $Z = \sum_{i=0}^{2} \exp(q_i[t]^\intercal k_i[t])$ giving us

$$\begin{aligned}
\frac{\partial(a_0[t]v_0[t])}{\partial h[t-1]} &= v_0[t](-\frac{\partial Z}{Z^2}) \\
&= \frac{-v_0[t]}{Z^2}\partial(e^{q_1[t]^\intercal k_1[t]}) \\
&= \frac{-v_0[t]}{Z^2}(k_1 Q_q + q_1^\intercal K_1)e^{q_1[t]^\intercal k_1[t]} \\
&= -v_0[t]\left(k_1 Q_q + q_1^\intercal K_1\right)\frac{e^{q_1[t]^\intercal k_1[t]}}{\left(\sum_{i=0}^{2}\exp(q_i[t]^\intercal k_i[t])\right)^2}
\end{aligned}$$

Similarly, we have that

$$\frac{\partial(a_2[t]v_2[t])}{\partial h[t-1]} = -v_2[t]\left(k_1 Q_q + q_1^\intercal K_1\right)\frac{e^{q_1[t]^\intercal k_1[t]}}{\left(\sum_{i=0}^{2}\exp(q_i[t]^\intercal k_i[t])\right)^2}$$

For $a_1[t]v_1[t]$, we use the product rule to find that

$$\begin{aligned}
\frac{\partial(a_1[t]v_1[t])}{\partial h[t-1]} &= v_1[t]\frac{\partial a_1[t]}{\partial h[t-1]} + a_1[t]\frac{\partial v_1[t]}{\partial h[t-1]} \\
\frac{\partial v_1[t]}{\partial h[t-1]} &= V_1 \\
\frac{\partial(a_1[t])}{\partial h[t-1]} &= \frac{\partial \exp(q_1[t]^\intercal k_1[t])}{Z} - \exp(q_1[t]^\intercal k_1[t])\frac{\partial Z}{Z^2} \\
&= \frac{(k_1 Q_1 + q_1^\intercal K_q)\exp(q_1[t]^\intercal k_1[t])}{Z} \\
&\quad - \exp(q_1[t]^\intercal k_1[t])\left(k_1 Q_q + q_1^\intercal K_1\right)\frac{e^{q_1[t]^\intercal k_1[t]}}{\left(\sum_{i=0}^{2}\exp(q_i[t]^\intercal k_i[t])\right)^2}
\end{aligned}$$

So finally, we have that

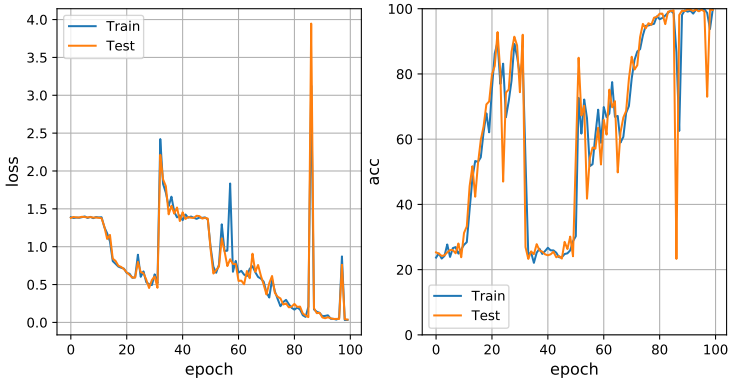
$$\begin{aligned}
\frac{\partial h[t]}{\partial h[t-1]} = & \\
& \frac{\partial(a_0[t]v_0[t])}{\partial h[t-1]} \\
& + v_1[t]\frac{\partial a_1[t]}{\partial h[t-1]} + a_1[t]\frac{\partial v_1[t]}{\partial h[t-1]} \\
& + \frac{\partial(a_2[t]v_2[t])}{\partial h[t-1]}
\end{aligned}$$

Each of which are worked out above.

(f) (2 pts) Suppose we know the derivative $\frac{\partial h[t]}{\partial h[T]}$, and $\frac{\partial \ell}{\partial h[t]}$ for all $t > T$. Please write down the expression for $\frac{\partial \ell}{\partial h[T]}$ for AttentionRNN(k). $\sum_{i=1}^{k}\frac{\partial h[T+i]}{\partial h[T]}\frac{\partial \ell}{\partial h[T+i]}$

1.3 Debugging loss curves (5pts)

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. (1pts) What caused the spikes on the left? exploding gradients
2. (1pts) How can they be higher than the initial value of the loss? many possible reasonable short answers - must allude to optimization instabilities
3. (1pts) What are some ways to fix them? gradient clipping
4. (2pts) Explain why the loss and accuracy are at these values before training starts. You may need to check the task definition in the notebook. accuracy is approx 1/4 if model is randomly initialized, loss is is just $\approx \log(4)$

2 Implementation (50pts + 7pts extra credit)

There are three notebooks in the practical part:

- (10 + 2pts) Mixture of Experts:hw2_moe.ipynb
- (20pts) Convolutional Neural Networks notebook: hw2_cnn.ipynb
- (15pts) 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.