Submission for Homework 2: Convolutional Neural Networks and Recurrent Neural Networks

Jay Nitin Kaoshik
jnk7726@nyu.edu
N13143537

CSCI-GA 2572 Deep Learning

Fall 2024

1 Theory (50pt)

1.1 Convolutional Neural Netoworks (15 pts)

(a) (1 pts) Given an input image of dimension 17×11, what will be output dimension after applying a convolution with 3×5 kernel, stride of 2, and no padding?

Answer for 1.1 (a):

We Know, $H = 17, W = 11$, $HK = 3, WK = 5$ and $S = 2$

Convolution Output Dimension =

$$\left\lfloor \frac{H - HK}{S} \right\rfloor + 1, \left\lfloor \frac{W - WK}{S} \right\rfloor + 1$$

Thus, Output = 8×4

(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 $HK \times WK$, padding P, stride S, dilation D, and F filters. Assume that $H \geq HK$, $W \geq WK$.

Answer for 1.1 (b):

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

(c) (12 pts) In this section, we are going to work with 1-dimensional convolutions. Let's consider an input $x[n] \in \mathbb{R}^7$, with $1 \le n \le 11$, e.g. it is a length 11 sequence with 7 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, $W \in \mathbb{R}^{1\times 7\times 3}$, 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: $f_W(x) \in \mathbb{R}^{42\times 42\times 42}$, $f_W(x)[i,j,k] = 42$.

Answer for 1.1 (c) (i):

$$f_W(x) \text{ Output Dimension } = \left\lfloor \frac{\text{Input Dimension} - \text{Kernel Size}}{\text{Stride}} \right\rfloor + 1$$

We Know, Input Dimension = 11, Kernel Size = 3, Stride = 2

$$f_W(x) \text{ Output Dimension } = 5$$

$$\implies f_W(x) \in \mathbb{R}^5$$

Now, the Input Sequence has a Length of 11 and 7 Channels making Each Position in Sequence a 7-Dimensional Vector. The Kernel has a Size of 3 which Will Allow the Kernel to Look at 3 Consecutive Positions in the Input Sequence for Each Output Calculation. It Operates over 7 Channels and Hence the Size is 7×3. Stride 2 Allows the Kernel to Jump 2 Positions in Input Sequence before Calculating Next Value.

For Each Position m in the Output ($1 \le m \le 5$), the Output Value in $f_W(x)[m]$ is Computed as the Sum of Element-Wise Products of Kernel and Corresponding Input Segment. For Output Position $m = 1$ the Kernel will Start at $n = 1$; for Output Position $m = 2$, Kernel will Start at $n = 3$; for Output Position $m = 3$, Kernel will Start at $n = 5$, etc ...

Thus, Kernel Starting Position $= 2 \times (m-1) + 1$
(2 Factor Comes from Stride and +1 Comes from Indexing Choice)

For Each Output Position m, Kernel Looks at 3 Consecutive Positions:

$$2 \times (m-1) + 1,\ 2 \times (m-1) + 2,\ 2 \times (m-1) + 3$$

Let's Assume Input Sequence at Position n and Channel c as $x[n][c]$. We Have Weights $W[c,i]$ for Each Channel c & Kernel Position i. Thus, the Value of the Output at Position m, denoted $f_W(x)[m]$, is the Sum of Element-Wise Products of Kernel Weights and the corresponding Input Values for the 7 Channels and 3 Consecutive positions.

$$f_W(x)[m]$$

$$= \sum_{c=1}^{7} x[2\times(m-1)+1][c]\cdot W[c,1]+x[2\times(m-1)+2][c]\cdot W[c,2]+x[2\times(m-1)+3][c]\cdot W[c,3]$$

$$= \sum_{c=1}^{7}\sum_{idx=1}^{3} x[2\times(m-1)+idx][c]\cdot W[c,idx]$$

Final Answer for 1.1 (c) (i):

$$f_W(x) \in \mathbb{R}^5$$

$$f_W(x)[m] = \sum_{c=1}^{7}\sum_{idx=1}^{3} x[2\times(m-1)+idx][c]\cdot W[c,idx]$$

(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 for 1.1 (c) (ii):

$$\text{Dimension of } \frac{\partial \boldsymbol{f_W(x)}}{\partial \boldsymbol{W}} \in \mathbb{R}^{5\times1\times7\times3}$$

$$\frac{\partial f_W(x)}{\partial W}[m,c,idx] = x[2\times(m-1)+idx][c]$$
where $1 \le m \le 5$, $1 \le c \le 7$, $1 \le idx \le 3$

(iii) (4 pts) What is the dimension of $\frac{\partial f_W(x)}{\partial x}$? Provide an expression for the values of the derivative $\frac{\partial f_W(x)}{\partial x}$.
Answer for 1.1 (c) (iii):

$$\text{Dimension of } \frac{\partial \boldsymbol{f_W(x)}}{\partial \boldsymbol{x}} \in \mathbb{R}^{5\times7\times11}$$

Now, We are Calculating the Gradient of $f_W(x)$ w.r.t. Input x. Each Output Position m is Influenced by Multiple Input Positions due to the Sliding Kernel. We Need to Thus Figure Out which Weights from the Kernel correspond to Which Positions in Input Sequence for a Given Output. Let's Assume m is the Output Position influenced by Input Position idx. We Know that $2\times(m-1)$ gives us the Starting Position in Input for an Output Position m. Thus $2\times(m-1)+idx$ adjusts the position in the input sequence relative to the starting point of the kernel for output position m. Similarly $idx-2\times(m-1)$ gives us the Relative Position of Input Element at idx within the Kernel Window at Output Position m. We Essentially Find where the Input Position idx falls within Kernel's Receptive Field at Output m. If $(idx-2\times(m-1)) \in (1,2,3)$ the Gradient Will Depend on the Weight else Gradient will be 0 because then idx won't Influence Output at m.

$$\frac{\partial f_W(x)}{\partial x}[m,c,idx] = W[c, idx - 2\times(m-1)]$$

$$\text{when } (idx - 2\times(m-1)) \in \{1,2,3\} \text{ else } \tfrac{\partial f_W(x)}{\partial x} = 0$$

$$\text{Here, } 1 \le m \le 5,\ 1 \le c \le 7,\ 1 \le idx \le 11$$

(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).

Answer for 1.1 (c) (iv):

$$\text{Dimension of } \frac{\partial \ell}{\partial \boldsymbol{W}} \in \mathbb{R}^{1\times 7\times 3}$$

$$\frac{\partial \ell}{\partial W} = \frac{\partial \ell}{\partial \boldsymbol{f_W(x)}} \times \frac{\partial \boldsymbol{f_W(x)}}{\partial \boldsymbol{W}}$$

We Know,

$$f_W(x)[m] = \sum_{c=1}^{7}\sum_{idx=1}^{3} x[2\times(m-1)+idx][c]\cdot W[c,idx]$$

$$\text{Thus, } \frac{\partial f_W(x)[m]}{\partial W[c,idx]} = x[2\times(m-1)+idx][c]$$

$$\frac{\partial \ell}{\partial W[c,idx]} = \sum_{m=1}^{5} \frac{\partial \ell}{\partial f_W(x)[m]} \cdot \frac{\partial f_W(x)[m]}{\partial W[c,idx]}$$

Hence,

$$\frac{\partial \ell}{\partial W[c,idx]} = \sum_{m=1}^{5} \frac{\partial \ell}{\partial f_W(x)[m]} \cdot x[2\times(m-1)+idx][c]$$

The expression in (i) shows the output of a convolution operation (forward pass) given some input sequence, channels and a fixed stride. The expression in (iv) is used to perform the backward pass to update gradients. Ironically, they are **both using the convolution operation!** In the forward pass, the stride causes the kernel to skip over input positions, meaning that the output at each position m depends on inputs spaced apart by the stride. In the backward pass, the gradient with respect to the weights must account for only the input positions that contributed to the output. This ensures that the gradient is correctly propagated back to the same input positions that influenced the

output. As a result, the backward pass can be interpreted as a convolution of the input gradient tensor with a version of the output gradient tensor that is effectively dilated, due to the spacing introduced by the stride during the forward pass. Reference (Page 34): https://deeplearning.cs.cmu.edu/F21/document/recitation/Recitation5/CNN_Backprop_Recitation_5_F21.pdf

1.2 Recurrent Neural Networks (30 pts)

1.2.1 Part 1

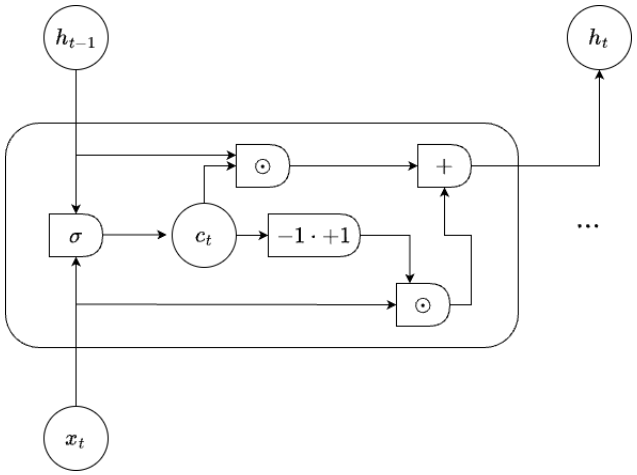
In this section we consider a simple recurrent neural network defined as follows:

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

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.

Answer for 1.2.1 (a):



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

Answer for 1.2.1 (b):

$$c[t] \in \mathbb{R}^m$$

(c) (3 pts) Derive expressions for $\frac{\partial h[t]}{\partial h[t-1]}$ and $\frac{\partial c[t]}{\partial h[t-1]}$

Answer for 1.2.1 (c):

$$c[t] = \sigma(W_c x[t] + W_h h[t-1])$$

$$\text{We Know, } \frac{\partial \sigma(a)}{\partial a} = \sigma(a) \odot (1 - \sigma(a))$$

$$\text{Thus, } \frac{\partial \boldsymbol{c[t]}}{\partial \boldsymbol{h[t-1]}} = \text{diag}(\sigma(W_c x[t] + W_h h[t-1]) \odot (1 - \sigma(W_c x[t] + W_h h[t-1]))) \cdot W_h$$

$$\text{Now, We Derive } \frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[t-1]}} \text{ knowing } h[t] = c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t]$$

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

$$\text{Thus, } \frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[t-1]}} = \text{diag}(c[t]) + \text{diag}(h[t-1]) \cdot \text{diag}(\sigma(W_c x[t] + W_h h[t-1]) \odot (1 - \sigma(W_c x[t] + W_h h[t-1]))) \cdot W_h$$
$$- \text{diag}(W_x x[t]) \cdot \text{diag}(\sigma(W_c x[t] + W_h h[t-1]) \odot (1 - \sigma(W_c x[t] + W_h h[t-1]))) \cdot W_h$$

(d) (4 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 the value of $\frac{\partial \ell}{\partial W_x}$ in terms of ℓ, h, W_x.
What are the similarities of backward pass and forward pass in this RNN?

Answer for 1.2.1 (d):

$$\text{Dimension of } \frac{\partial \ell}{\partial \boldsymbol{W_x}} \in \mathbb{R}^{n \times m}$$

$$\frac{\partial \ell}{\partial \boldsymbol{W_x}} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial \boldsymbol{h[t]}} \cdot \frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{W_x}}$$

$$\text{We Know, } h[t] = c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t]$$

$$\frac{\partial \ell}{\partial \boldsymbol{W_x}} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial \boldsymbol{h[t]}} \cdot (1 - c[t]) \odot x[t]$$

Although in BPTT, $h[t]$ depends on both $h[t-1]$ and W_x and thus we need to maintain the dependence on previous time steps. The additional term which is to be included is shown below. Reference: https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

$$\text{Additional Term} = \sum_{i=1}^{t-1} \left(\Pi_{j=i+1}^{t} \frac{\partial \boldsymbol{h[j]}}{\partial \boldsymbol{h[j-1]}} \right) (1 - c[i]) \odot x[i]$$

$$\text{Thus, } \frac{\partial \ell}{\partial \boldsymbol{W_x}} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial \boldsymbol{h[t]}} \cdot \left[(1 - c[t]) \odot x[t] + \sum_{i=1}^{t-1} \left(\Pi_{j=i+1}^{t} \frac{\partial \boldsymbol{h[j]}}{\partial \boldsymbol{h[j-1]}} \right) (1 - c[i]) \odot x[i] \right]$$

Minor Note: The $x[t]$ and $x[i]$ terms above have to be converted to a shape of $\mathbb{R}^{m\times n\times m}$ to facilitate multiplication with the $(1-c[t])$ or $(1-c[i])$ term. 1 of the columns in the converted representation will hold the values $x[t]$ or $x[i]$.

Similarity in Forward & Backward Similar to 1.1 where we had convolution operation in both forward and backward passes of CNNs, we now have recurrences in both forward and backward passes of RNNs!

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

Answer for 1.2.1 (e):

Yes the above RNN holds the risk of vanishing gradients. This is because at every recurrent step $h[t-1]$ is multiplied with $c[t]$, which in itself is a sigmoid output ranging from 0 to 1. For exploding gradients we refer to our calculation of $\frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[t-1]}}$ in 1.2.1 (c). There will only be a possibility of gradient explosion if W terms get abnormally large. Although at every recurrent step, we dampen the effect from those values using the sigmoid function.

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

Answer for 1.2.2 (a):

$$a[t] \in \mathbb{R}^3$$

(b) (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.

Answer for 1.2.2 (b):

$$
\begin{aligned}
q_0[t], q_1[t], \ldots, q_k[t] &= Q_0 x[t], Q_1 h[t-1], \ldots, Q_k h[t-k] \\
k_0[t], k_1[t], \ldots, k_k[t] &= K_0 x[t], K_1 h[t-1], \ldots, K_k h[t-k] \\
v_0[t], v_1[t], \ldots, v_k[t] &= V_0 x[t], V_1 h[t-1], \ldots, V_k h[t-k] \\
w_i[t] &= q_i[t]^\top k_i[t], \quad \text{for } i = 0, 1, \ldots, k \\
a[t] &= \text{softargmax}(w_0[t], w_1[t], \ldots, w_k[t]) \\
h[t] &= \sum_{i=0}^{k} a_i[t] v_i[t]
\end{aligned}
$$

(c) (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. How many weight matrices are required for this model? Is there a way to limit this number? HINT: We can do this by tying together some set of parameters, e.g. weight sharing.

Answer for 1.2.2 (c):

$$
\begin{aligned}
q_0[t], q_1[t], \ldots, q_\infty[t] &= Q_0 x[t], Q_1 h[t-1], \ldots, Q_\infty h[t-\infty] \\
k_0[t], k_1[t], \ldots, k_\infty[t] &= K_0 x[t], K_1 h[t-1], \ldots, K_\infty h[t-\infty] \\
v_0[t], v_1[t], \ldots, v_\infty[t] &= V_0 x[t], V_1 h[t-1], \ldots, V_\infty h[t-\infty] \\
w_i[t] &= q_i[t]^\top k_i[t], \quad \text{for } i = 0, 1, \ldots, \infty \\
a[t] &= \text{softargmax}(w_0[t], w_1[t], \ldots, w_\infty[t]) \\
h[t] &= \sum_{i=0}^{\infty} a_i[t] v_i[t]
\end{aligned}
$$

So we would require ∞ matrices to actually produce a AttentionRNN(∞). Without any parameter sharing, we would need infinitely many weight matrices Q_i, K_i and V_i for each past state vector $h[t-1]$, $h[t-2]$, ... Clearly, this is impractical. A practical approach is to use the same set of weight matrices for arbitrarily set π consecutive past state vectors. Thus, $q_{i+j}[t] \propto Q_j[t]$ where $i \in \{1, 2, ...\pi\}$ and j is the iterator over the large number of past state vectors we use. This will hence be replicated for $k_{i+j}[t]$and $v_{i+j}[t]$. A similar approach has been shown for Transformers in the paper Xiao, Tong, et al. "Sharing attention weights for fast transformer." arXiv preprint arXiv:1906.11024 (2019) https://arxiv.org/abs/1906.11024.

(d) (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).

Answer for 1.2.2 (d):

We have the equations for $h[t]$ and the attention weights:

$$h[t] = a_0(t) \cdot v_0(t) + a_1(t) \cdot v_1(t) + a_2(t) \cdot v_2(t)$$

Where the attention weights are:

$$a_0(t) = \frac{e^{w_0}}{e^{w_0} + e^{w_1} + e^{w_2}}, \quad a_1(t) = \frac{e^{w_1}}{e^{w_0} + e^{w_1} + e^{w_2}}, \quad a_2(t) = \frac{e^{w_2}}{e^{w_0} + e^{w_1} + e^{w_2}}$$

The weight terms w_0, w_1, w_2 are:

$$w_0 = q_0(t)^\top K_0(t) = Q_0 x[t]^\top K_0 x[t]$$

$$w_1 = q_1(t)^\top K_1(t) = Q_1 h[t-1]^\top K_1 h[t-1]$$

$$w_2 = q_2(t)^\top K_2(t) = Q_2 h[t-2]^\top K_2 h[t-2]$$

For the values $v_0(t), v_1(t), v_2(t)$, we have:

$$v_0(t) = V_0 x[t], \quad v_1(t) = V_1 h[t-1], \quad v_2(t) = V_2 h[t-2]$$

Computing the Gradient $\frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[t-1]}}$ using Direct Dependence Term of $a_1(t) \cdot v_1(t)$ as it directly depends on $h[t-1]$.

$$\frac{\partial (a_1(t) \cdot \boldsymbol{v_1(t)})}{\partial \boldsymbol{h[t-1]}} = a_1(t) \cdot V_1 + v_1(t) \cdot \frac{\partial a_1(t)}{\partial \boldsymbol{h[t-1]}}$$

Now We Consider Indirect Dependence through Attention Weights. The weights $a_0(t), a_1(t), a_2(t)$ depend on $w_1(t)$ which depends on $h[t-1]$.

Chain Rule for Derivative of $a_0(t)$ with respect to $h[t-1]$,

$$\frac{\partial a_0(t)}{\partial \boldsymbol{h[t-1]}} = \frac{\partial a_0(t)}{\partial w_1(t)} \cdot \frac{\partial w_1(t)}{\partial \boldsymbol{h[t-1]}}$$

Derivative of $a_1(t)$,

$$\frac{\partial a_1(t)}{\partial \boldsymbol{h[t-1]}} = \frac{\partial a_1(t)}{\partial w_1(t)} \cdot \frac{\partial w_1(t)}{\partial \boldsymbol{h[t-1]}}$$

Similarly for $a_2(t)$,

$$\frac{\partial a_2(t)}{\partial \boldsymbol{h[t-1]}} = \frac{\partial a_2(t)}{\partial w_1(t)} \cdot \frac{\partial w_1(t)}{\partial \boldsymbol{h[t-1]}}$$

The final gradient will be:

$$\frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[t-1]}} = a_1(t) \cdot V_1 + v_1(t) \cdot \frac{\partial a_1(t)}{\partial \boldsymbol{h[t-1]}} + v_0(t) \cdot \frac{\partial a_0(t)}{\partial \boldsymbol{h[t-1]}} + v_2(t) \cdot \frac{\partial a_2(t)}{\partial \boldsymbol{h[t-1]}}$$

To calculate the derivatives $\frac{\partial a_0(t)}{\partial h[t-1]}, \frac{\partial a_1(t)}{\partial h[t-1]}, \frac{\partial a_2(t)}{\partial h[t-1]}$, we will apply the chain rule shown above and use the expressions for $a_0(t), a_1(t), a_2(t)$ and w_0, w_1, w_2.

1. $\frac{\partial a_0(t)}{\partial \boldsymbol{h[t-1]}}$

The expression for $a_0(t)$ is:

$$a_0(t) = \frac{e^{w_0}}{e^{w_0} + e^{w_1} + e^{w_2}}$$

Taking the partial derivative of $a_0(t)$ with respect to w_1:

$$\frac{\partial a_0(t)}{\partial w_1(t)} = -a_0(t) \cdot a_1(t)$$

The partial derivative of w_1 with respect to $h[t-1]$ is:

$$\frac{\partial w_1(t)}{\partial \boldsymbol{h[t-1]}} = 2 \cdot Q_1 h[t-1]^\top K_1$$

Thus:

$$\frac{\partial a_0(t)}{\partial \boldsymbol{h[t-1]}} = -a_0(t) \cdot a_1(t) \cdot 2 \cdot Q_1 h[t-1]^\top K_1$$

2. $\frac{\partial a_1(t)}{\partial \boldsymbol{h[t-1]}}$

The expression for $a_1(t)$ is:

$$a_1(t) = \frac{e^{w_1}}{e^{w_0} + e^{w_1} + e^{w_2}}$$

Taking the partial derivative of $a_1(t)$ with respect to w_1:

$$\frac{\partial a_1(t)}{\partial w_1(t)} = a_1(t) \cdot (1 - a_1(t))$$

Thus:

$$\frac{\partial a_1(t)}{\partial \boldsymbol{h[t-1]}} = a_1(t) \cdot (1 - a_1(t)) \cdot 2 \cdot Q_1 h[t-1]^\top K_1$$

3. $\frac{\partial a_2(t)}{\partial \boldsymbol{h[t-1]}}$

The expression for $a_2(t)$ is:

$$a_2(t) = \frac{e^{w_2}}{e^{w_0} + e^{w_1} + e^{w_2}}$$

Taking the partial derivative of $a_2(t)$ with respect to w_1:

$$\frac{\partial a_2(t)}{\partial w_1(t)} = -a_2(t) \cdot a_1(t)$$

Thus:

$$\frac{\partial a_2(t)}{\partial \boldsymbol{h[t-1]}} = -a_2(t) \cdot a_1(t) \cdot 2 \cdot Q_1 h[t-1]^\top K_1$$

Now, combining the direct and indirect dependencies:

$$\frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[t-1]}} = a_1(t)\cdot V_1 + \left(\frac{\partial a_1(t)}{\partial \boldsymbol{h[t-1]}}\right)\cdot v_1(t) + \left(\frac{\partial a_0(t)}{\partial \boldsymbol{h[t-1]}}\right)\cdot v_0(t) + \left(\frac{\partial a_2(t)}{\partial \boldsymbol{h[t-1]}}\right)\cdot v_2(t)$$

$$\begin{aligned}\frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[t-1]}} &= a_1(t)\cdot V_1 + \left(a_1(t)\cdot(1-a_1(t))\cdot 2\cdot Q_1 h[t-1]^\top K_1\right)\cdot v_1(t) \\ &\quad + \left(-a_0(t)\cdot a_1(t)\cdot 2\cdot Q_1 h[t-1]^\top K_1\right)\cdot v_0(t) \\ &\quad + \left(-a_2(t)\cdot a_1(t)\cdot 2\cdot Q_1 h[t-1]^\top K_1\right)\cdot v_2(t)\end{aligned}$$

(e) (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).

Answer for 1.2.2 (e):

Given that we know the derivatives $\frac{\partial h[t]}{\partial h[T]}$ and $\frac{\partial \ell}{\partial h[t]}$ for all $t > T$, we can express the derivative $\frac{\partial \ell}{\partial h[T]}$ for AttentionRNN(k) as follows:

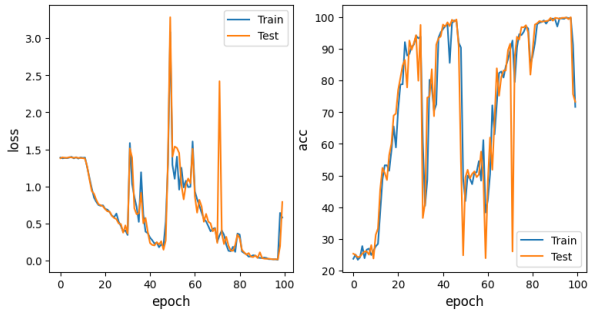
We can express $\frac{\partial \ell}{\partial h[T]}$ by summing over the contributions of future time steps $t > T$:

$$\frac{\partial \ell}{\partial \boldsymbol{h[T]}} = \sum_{t=T+1}^{K} \frac{\partial \ell}{\partial \boldsymbol{h[t]}} \cdot \frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[T]}}$$

The system of equations for AttentionRNN(k) involves $q_i[t], k_i[t], v_i[t], w_i[t]$, $a_i[t], h[t]$, which are calculated based on previous time steps up to $h[t-k]$. Therefore, $\frac{\partial \boldsymbol{h[t]}}{\partial \boldsymbol{h[T]}}$ includes contributions from attention weights and previous hidden states through recursive dependencies as shown in 1.2.2 (d). Each term $h[t]$ depends on $h[T]$ via attention weights $a_i[t]$ and values $v_i[t]$.

1.3 Debugging loss curves (5pts)

When we run the notebook *seq_classification.ipynb* (included in hw, and we also went through this code in class) we saw RNN training curves. In Section 8 "Visualize LSTM", we observed some "kinks" in the loss curve.



(a) (2pts) What caused the spikes on the left?

Answer for 1.3 (a):

The main reason for the spikes is the **Exploding Gradient** problem inherent in RNNs. This happens when the gradients become excessively large and lead to unstable updates, resulting in abrupt spikes in the loss.

(b) (1pts) How can they be higher than the initial value of the loss?

Answer for 1.3 (b):

During gradient explosion, the gradients can overshoot to abnormally large values. The loss printed initially is due to the random initialization of weights depending on the setup. Thus, it will have no relation or bounds for when the gradients accumulate to large values and cross initial loss.

(c) (1pts) What are some ways to fix them?

Answer for 1.3 (c):

Gradient Clipping (Norm Based or Layer Based Value), Xavier/He Initialization of Weights, Normalization Layers, Dynamic Learning Rate Optimizers like Adam, Weight Decay, Reducing Learning Rate

(d) (1pts) Explain why the accuracy starts at that certain number in the start of training. You may need to check the task definition in the notebook.

Answer for 1.3 (d):

The accuracy starts from around 25% which makes sense when we see the task definition of the classification where we have 4 sequence classes Q, R, S and U. The initial prediction is as good as a randomized prediction of 1/4.

2 Implementation (50pts)

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) : This builds on Section 1.3 of the theoretical part.
  - Change the model training procedure of Section 8 in *seq_classification.ipynb* 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.

We encourage you to upload the notebooks to Google Colab to work on it.
First two notebooks contain parts marked as `TODO`, where you should put your code. After you finish, you should download your solutions, and then include them as a part of your submission zip.