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

Chuanyang Jin

February 26, 2023

1.1 Convolutional Neural Netoworks (15 pts)

(a) (1 pts) Given an input image of dimension 12×21, what will be output dimension after applying a convolution with 5×4 kernel, stride of 4, and no padding?

Solution.

$\tilde{H} = \lfloor (12-5)/4 \rfloor + 1 = 2$
$\tilde{W} = \lfloor (21-4)/4 \rfloor + 1 = 5$
Output dimension: 2×5. □

(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 $H \geq K$, $W \geq K$.

Solution.

$\tilde{H} = \lfloor (H + 2P - D(K-1) - 1)/S \rfloor + 1$
$\tilde{W} = \lfloor (W + 2P - D(K-1) - 1)/S \rfloor + 1$
Number of filters $= F$
Output dimension: $F \times (\lfloor (H + 2P - D(K-1) - 1)/S \rfloor + 1) \times (\lfloor (W + 2P - D(K-1) - 1)/S \rfloor + 1)$. □

(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, $W \in \mathbb{R}^{1\times5\times3}$, 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: $f_W(x) \in \mathbb{R}^{42\times42\times42}$, $f_W(x)[i,j,k] = 42$.

Solution.

The output length is $\lfloor \frac{7-3}{2} \rfloor + 1 = 3$. Since the input has 5 channels and there is 1 filter, the output will have 1 channel. Therefore, $f_W(x) \in \mathbb{R}^{3\times1}$.

The value of each element of the output $f_W(x)$ can be computed by:

$$f_W(x)[i,1] = \sum_{c=1}^{5} \sum_{m=-1}^{1} W\left[1, c, m+2\right] x\left[2i+m, c\right]$$

□

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

Solution.

$\frac{\partial f_W(x)}{\partial W} \in \mathbb{R}^{3\times1\times5\times3}$.

Its values can be expressed by

$$\begin{aligned}\frac{\partial f_W(x)}{\partial W}[i,1,c,n] &= \frac{\partial f_W(x)}{\partial W}[i,1,c,m+2] \\ &= x[2i+m,c] \\ &= x[2i+n-2,c]\end{aligned}$$

where $i \in \{1,2,3\}$, $c \in \{1,2,3,4,5\}$, $n \in \{1,2,3\}$, $i \in \{-1,0,1\}$. □

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

Solution.

$\frac{\partial f_W(x)}{\partial x} \in \mathbb{R}^{3\times1\times7\times5}$.

Its values can be expressed by

$$\frac{\partial f_W(x)}{\partial W}[i,1,j,c] = \begin{cases} W[1,c,j-2i+2] & -1 \leq j-2i \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

□

(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).

Solution.

$\frac{\partial l}{\partial W} \in \mathbb{R}^{1\times1\times5\times3}$.

Its values can be expressed by

$$\frac{\partial l}{\partial W}[1,1,c,n] = \sum_{i=1}^{3} \frac{\partial l}{\partial f_W(x)[i]} x\,[2i+n-2,c]$$
$$= \sum_{i=1}^{3} \frac{\partial l}{\partial f_W(x)}\,[1,i]\,x\,[2i+n-2,c]$$

The two results share a similar form, since they both include the use of convolution operations on the sequence x.

However, there is a notable distinction between the two expressions. In the expression in (i), the weights from the convolutional layer are applied to the elements of x to generate the output. In contrast, this expression involves multiplication of the elements of x with derivatives to derive the gradient of the loss. □

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:

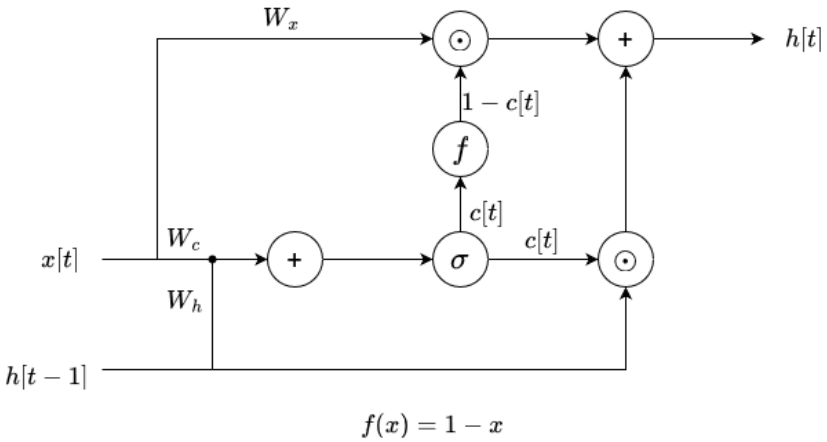
$$c[t] = \sigma(W_c x[t] + W_h h[t-1])$$

$$h[t] = c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t]$$

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, $h[0] = 0$.

(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.

Solution.



□

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

Solution.

Since $W_c \in \mathbb{R}^{m \times n}, x[t] \in \mathbb{R}^n$, we know $W_c x[t] \in \mathbb{R}^m$. Then $c[t] \in \mathbb{R}^m$. □

(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?

Solution.

Using the chain rule, we have:

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

$\frac{\partial \ell}{\partial h[t]}$ is given. To compute $\frac{\partial h[t]}{\partial W_x}$, we can use the expression for $h[t]$:

$$h[t] = c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t]$$

Therefore

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

The dimensions of $\frac{\partial \ell}{\partial W_x}$ is $R^{n \times m}$.

The backward pass and forward pass are similar in the sense that the backward pass replaces the weights W_x in the forward pass with the derivative $\frac{\partial \ell}{\partial h[t]}$ to get the gradients of weights $\frac{\partial \ell}{\partial W_x}$. □

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

Solution.

Yes, the simple recurrent neural network can be subject to vanishing or exploding gradients.

The gradients are computed through backpropagation through time, which involves computing the chain rule of derivatives over many time steps. For example, the gradient of $h[t]$ with respect to $h[t-1]$ depends on the value of $c[t]$. If $c[t]$ is close to 0, the gradient will be small, which can lead to vanishing gradients. On the other hand, if $c[t]$ is close to 1, the gradient will be large, which can lead to exploding gradients.

The same reasoning can be applied to W_h since it is also involved in the computation of $c[t]$. Thus, the gradient of the loss function with respect to these parameters can also suffer from vanishing or exploding gradients. □

1.2.2 Part 2

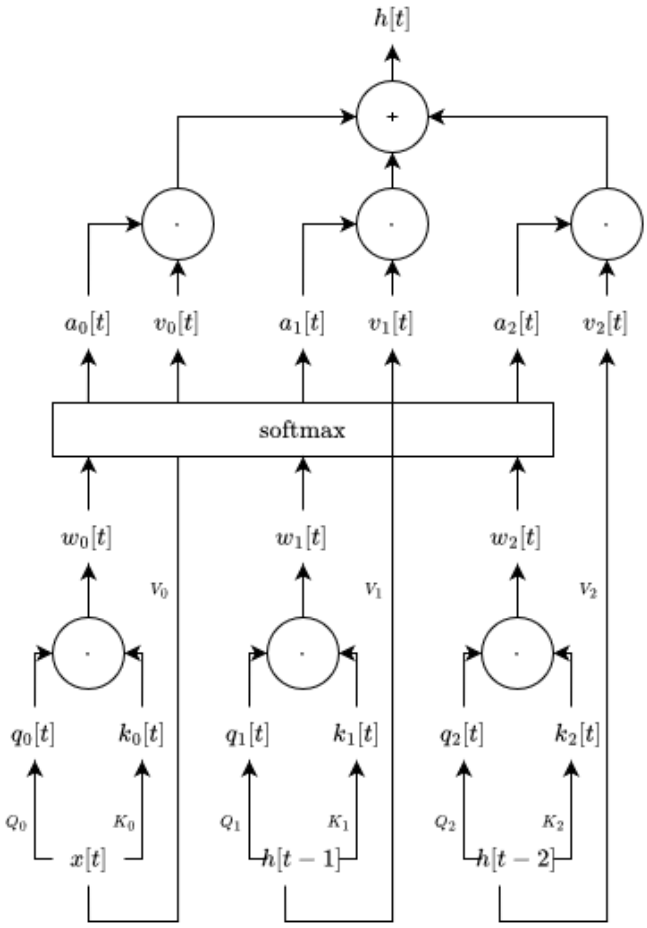
We define an AttentionRNN(2) as

$$\begin{aligned}
q_0[t], q_1[t], q_2[t] &= Q_0 x[t], Q_1 h[t-1], Q_2 h[t-2] \\
k_0[t], k_1[t], k_2[t] &= K_0 x[t], K_1 h[t-1], K_2 h[t-2] \\
v_0[t], v_1[t], v_2[t] &= V_0 x[t], V_1 h[t-1], V_2 h[t-2] \\
w_i[t] &= q_i[t]^T k_i[t] \\
a[t] &= \text{softargmax}(w_0[t], w_1[t], w_2[t]) \\
h[t] &= \sum_{i=0}^{2} a_i[t] v_i[t]
\end{aligned}$$

where $x[t], h[t] \in \mathbb{R}^n$, and $Q_i, K_i, V_i \in \mathbb{R}^{n \times n}$. We define $h[t] = 0$ 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.

Solution.



□

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

Solution.

Since $x[t], h[t] \in \mathbb{R}^n$, and $Q_i, K_i, V_i \in \mathbb{R}^{n \times n}$, we know $q_i[t], k_i[t], v_i[t], w_i[t] \in \mathbb{R}^n$. Then $a[t] \in \mathbb{R}^3$. □

(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.

Solution.

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

where $x[t], h[t] \in \mathbb{R}^n$, and $Q_i, K_i, V_i \in \mathbb{R}^{n \times n}$. □

(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.

Solution.

$$
\begin{aligned}
q_0[t], q_1[t], ..., q_t[t] &= Q_x x[t], Qh[t-1], ..., Qh[0] \\
k_0[t], k_1[t], ..., k_t[t] &= K_x x[t], Kh[t-1], ..., Kh[0] \\
v_0[t], v_1[t], ..., v_t[t] &= V_x x[t], Vh[t-1], ..., Vh[0] \\
w_i[t] &= q_i[t]^T k_i[t] \\
a[t] &= \text{softargmax}(w_0[t], w_1[t], ..., w_t[t]) \\
h[t] &= \sum_{i=0}^{t} a_i[t] v_i[t]
\end{aligned}
$$

where $x[t], h[t] \in \mathbb{R}^n$, and $Q, K, V \in \mathbb{R}^{n \times n}$. □

(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).

Solution.

To compute $\frac{\partial h[t]}{\partial h[t-1]}$ for AttentionRNN(2), we can use the chain rule of differentiation and write:

$$
\begin{aligned}
\frac{\partial h[t]}{\partial h[t-1]} &= \frac{\partial}{\partial h[t-1]} \left(\sum_{i=0}^{2} a_i[t] v_i[t] \right) \\
&= \sum_{i=0}^{2} \frac{\partial a_i[t]}{\partial h[t-1]} v_i[t] + \sum_{i=0}^{2} a_i[t] \frac{\partial v_i[t]}{\partial h[t-1]}
\end{aligned}
$$

Using the given expressions, we can write:

$$\begin{aligned}
w_0[t] &= q_0[t]^T k_0[t] \\
&= x[t]^T Q_0^T K_0 x[t] \\
w_1[t] &= q_1[t]^T k_1[t] \\
&= h[t-1]^T Q_1^T K_1 h[t-1] \\
w_2[t] &= q_2[t]^T k_2[t] \\
&= h[t-2]^T Q_2^T K_2 h[t-2] \\
a_i[t] &= \frac{\exp(w_i[t])}{\sum_{j=0}^{2} \exp(w_j[t])} \\
h[t] &= \sum_{i=0}^{2} a_i[t] v_i[t]
\end{aligned}$$

Now we can compute the derivatives:

When $i = 1$,

$$\begin{aligned}
\frac{\partial a_i[t]}{\partial h[t-1]} &= \frac{\partial a_1[t]}{\partial w_1[t]} \frac{\partial w_1[t]}{\partial h[t-1]} \\
&= a_1[t](1 - a_1[t]) \frac{\partial w_1[t]}{\partial h[t-1]} \\
\frac{\partial v_i[t]}{\partial h[t-1]} &= V_1^T
\end{aligned}$$

When $i = 0, 2$,

$$\begin{aligned}
\frac{\partial a_i[t]}{\partial h[t-1]} &= \frac{\partial a_i[t]}{\partial w_1[t]} \frac{\partial w_1[t]}{\partial h[t-1]} \\
&= -a_i[t] a_1[t] \frac{\partial w_1[t]}{\partial h[t-1]} \\
\frac{\partial v_i[t]}{\partial h[t-1]} &= 0
\end{aligned}$$

Where

$$\begin{aligned}
\frac{\partial w_1[t]}{\partial h[t-1]} &= \frac{\partial q_1[t]}{\partial h[t-1]} k_1[t] + \frac{\partial k_1[t]}{\partial h[t-1]} q_1[t] \\
&= Q^T k_1[t] + K^T q_1[t]
\end{aligned}$$

Therefore

$$\begin{aligned}
\frac{\partial h[t]}{\partial h[t-1]} &= a_1[t] V_1^T + \sum_{i=0}^{2} a_1[t] \frac{\partial w_1[t]}{\partial h[t-1]} v_i[t] - \sum_{i=0}^{2} a_i[t] a_1[t] \frac{\partial w_1[t]}{\partial h[t-1]} v_i[t] \\
&= a_1[t] V_1^T + \sum_{i=0}^{2} (1 - a_i[t]) a_1[t] (Q^T k_1[t] + K^T q_1[t]) v_i[t]
\end{aligned}$$

□

(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).

Solution.

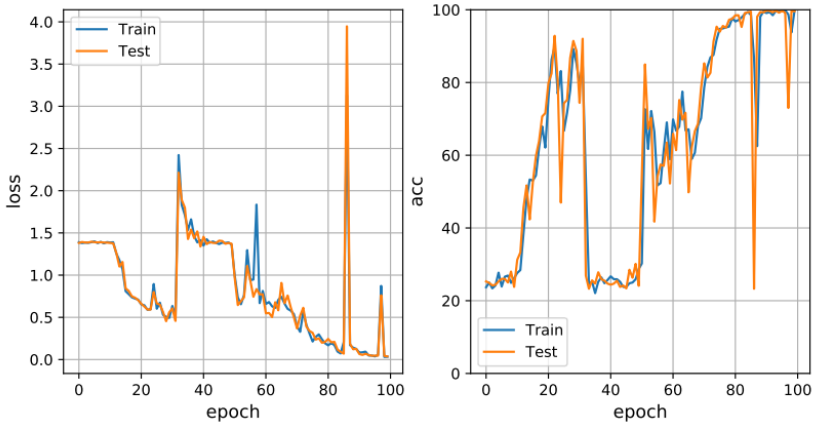
Assume the length of the sequence is L, using the chain rule of differentiation, we can write:

$$\frac{\partial \ell}{\partial h[T]} = \sum_{t=T+1}^{L-1} \frac{\partial \ell}{\partial h[t]} \frac{\partial h[t]}{\partial h[T]}$$

□

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?

   Solution.

   Gradient explosion, which causes the model parameters to update in very large steps. □

2. (1pts) How can they be higher than the initial value of the loss?

   Solution.

   The gradient of the loss function with respect to the model parameters becomes too large, so the model parameters can be updated with very large values, leading to a skyrocketing of the value of the loss which can be higher than the initial value of the loss. □

3. (1pts) What are some ways to fix them?

   Solution.

   We can use gradient clipping, which scales down the gradients if their norm exceeds a certain threshold. This approach limits the magnitude of the gradients and prevents them from becoming too large.

   We can also reduce the learning rate or increase the batch size. □

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.

Solution.

The initial cross entropy loss is about 1.4 and the initial accuracy is about 25%. Before the training starts, the model almost predict randomly among the four classes. So the loss and accuracy can be estimated by:

$$\mathcal{L} \approx -\sum_{i=1}^{4} \frac{1}{4} \log(\frac{1}{4}) = -\log(\frac{1}{4}) = \log(4) = 1.386$$

$$\text{acc} \approx \frac{1}{4} = 0.25$$

□