CSCI-GA 2572 Deep Learning - Homework 2

Due: October 1 @ 11:59pm

Name: Stephen Spivack (ss7726)

1 Theory

1.1 Convolutional neural networks

(a) Given an input image of dimension 11 × 19, what will be output dimension after applying a convolution with 5×4 kernel, stride of 4, and no padding?

We can apply this formula to each dimension of the image with respect to the kernel:

$dimensions_{out} = ((dimension_{input} - dimension_{kernel})/stride) + 1$

which, in our case, results in an output dimensionality of 2 x 4 (when rounding down to the nearest pixel).

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

Without dilation:

Output Height $= ((H - K + 2 * P)/S) + 1$

Output Width $= ((W - K + 2 * P)/S) + 1$

where the number of filters is F

With dilation:

Output Height $= ((H - (K - 1) * D + 2 * P)/S) + 1$

Output Width $= ((W - (K - 1) * D + 2 * P)/S) + 1$

where the number of filters is F

This gives us the final output dimension:

$((H - (K - 1) * D + 2 * P)/S + 1) \times ((W - (K - 1) * D + 2 * P)/S + 1) \times F$

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

$dimension_{out} = ((dimension_{input} - dimension_{kernel})/stride) + 1 = ((7-3)/2) + 1 = 3$

Given that there are 5 channels, we have:

$f_W(x) \in \mathbb{R}^{3\times5}$

with convolutional layer output:

$f_W(x)[i,j] = \sum_{m=1}^{3} x[2i+m-j] \cdot W[0,j,m]$

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

Same shape as the weights: $\frac{\partial f_W(x)}{\partial W} \in \mathbb{R}^{1\times5\times3}$

with the following expression for the derivative with respect to the weights:

$\frac{\partial f_W(x)}{\partial W}[0,j,m] = \sum_{i=1}^{3} x[2i+m-j] \cdot \frac{\partial l}{\partial f_W(x)}[i,j]$

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

Same shape as the inputs: $\frac{\partial f_w(x)}{\partial x} \in \mathbb{R}^{7\times5}$

with the following expression for the derivative with respect to the inputs:

$\frac{\partial f_W(x)}{\partial x}[n,c] = \sum_{k=1}^{3} \frac{\partial l}{\partial f_W(x)}[n-2k,c] \cdot W[0,c,k]$

(iv) Now, suppose you are given the gradient of the loss l with respect to the output of the convolutional layer $f_w(x)$, i.e., $\frac{\partial l}{\partial f_w(x)}$. What is the dimension of $\frac{\partial l}{\partial W}$? Provide an expression for $\frac{\partial l}{\partial W}$. Explain similarities and differences of this expression and the expression in (i).

Same shape as the weights (same as (ii)): $\frac{\partial l}{\partial W} \in \mathbb{R}^{1\times5\times3}$

with the following expression:

$\frac{\partial l}{\partial W}[0,j,m] = \sum_{i=1}^{3} x[2i+m-j] \cdot \frac{\partial l}{\partial f_W(x)}[i,j]$

Key similarity is that both use convolution between input and set of weights; key difference is that (i) computes output feature map whereas (iv) computes gradient of the loss with respect to filter weights

1.2 Recurrent neural networks

1.2.1 Part 1

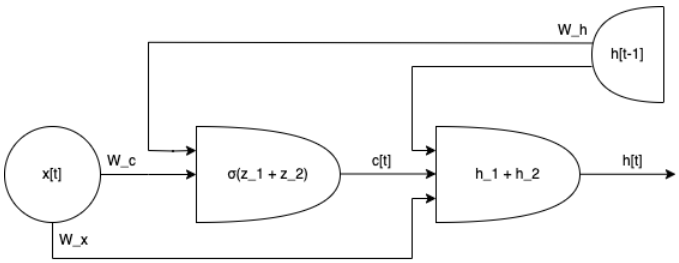
In this section we consider a simple recurrent neural network defined as follows:

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

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

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) Draw a diagram for this recurrent neural network.



z_1 = W_c X[t]
z_2 = W_h h[t-1]
h_1 = c[t] ∘ h[t-1]
h_2 = (1-c[t]) ∘ W_x X[t]

At each time step t, $x[t]$ is the input; the network maintains a hidden state vector $h[t]$; $c[t]$ acts as a gate that controls how much of the previous hidden state $h[t-1]$ is preserved in the current hidden state $h[t]$.

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

The dimension of $c[t]$ is m.

(c) Suppose that we run the RNN to get a sequence of $h[t]$ for t from 1 to K. Assuming we know the derivative $\frac{\partial l}{\partial h[t]}$ provide dimension of and an expression for values of $\frac{l}{W_x}$. What are the similarities of backward pass and forward pass in this RNN?

$\frac{l}{W_x} = \frac{\partial l}{\partial h[t]} \frac{\partial h[t]}{\partial W_x}$

From our RNN equations above we have:

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

Therefore, the dimension of $\frac{l}{W_x}$ is the same as W_x, which is $\mathbb{R}^{m \times n}$.

Both of them involve iterating through each time step. In the forward pass, we compute $c[t]$ and $h[t]$ directly, whereas in the backward pass we compute the gradients of $c[t]$ and $h[t]$.

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

Potential for vanishing gradient: presense of sigmoid activation in $c[t]$

Potential for exploding gradient: element-wise multiplication of the complement of the gate and the weight W_x in the update equation for the hidden state.

1.2.2 Part 2

We define an AttentionRNN(2) as

$q_0[t], q_1[t], q_2[t] = Q_0 x[t], Q_1 h[t-1], Q_2 h[t-2]$

$k_0[t], k_1[t], k_2[t] = K_0 x[t], K_1 h[t-1], K_2 h[t-2]$

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

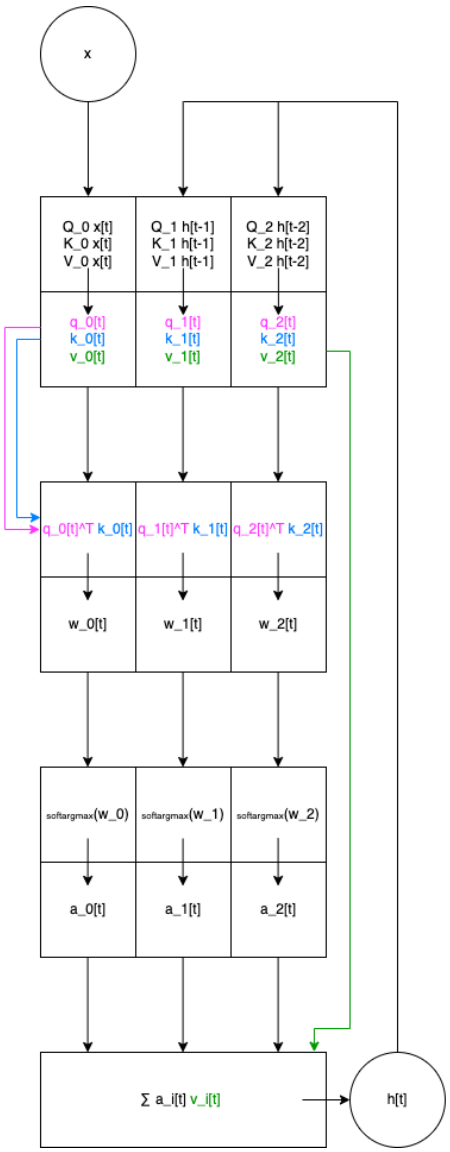
$w_i[t] = q_i[t]^T k_i[t]$

$a[t] = softargmax([w_o[t], w_1[t], w_2[t]])$

$h[t] = \sum_{i=1}^{2} a_i[t] v_i[t]$

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 $t < 1$.

(a) Draw a diagram for this recurrent neural network



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

The dimension is 3, and in general is k for a given $AttentionRNN(k)$ network

(c) Extend this to, AttentionRNN(k), a network that uses the last k state vectors h. Write out the system of equations that defines it.

In this setting the system of equations can be written as:

For $i = 0$ to $k - 1$:

$q_i[t] = Q_i x[t]$

$k_i[t] = K_i h[t-1]$

$v_i[t] = V_i h[t-i]$

$w_i[t] = q_i[t]^T k_i[t]$

With the following attention weights:

$w_0[t], w_1[t], ..., w_{k-1}[t]$

And the following output:

$h[t] = \sum_{i=0}^{i=k-1} a_i[t] v_i[t]$

(d) 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.

Now the system of equations can be written as:

For any $i \geq 0$:

$q_i[t] = Q_i x[t]$

$k_i[t] = K_i h[t-1]$

$v_i[t] = V_i h[t-i]$

$w_i[t] = q_i[t]^T k_i[t]$

With the following attention weights:

$w_0[t], w_1[t], w_2[t], \ldots$

And the following output:

$h[t] = \sum_{i=0}^{\infty} a_i[t] v_i[t]$

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

$\frac{\partial h[t]}{\partial h[t-1]} = \frac{\partial}{\partial h[t-1]} \left(\sum_{i=0}^{1} a_i[t] v_i[t] \right)$

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

$\frac{\partial h[t]}{\partial h[t-1]} = a_0[t] \frac{\partial v_0[t]}{\partial h[t-1]} + a_1[t] \frac{\partial v_1[t]}{\partial h[t-1]}$

(f) Suppose we know the derivative $\frac{\partial h[t]}{\partial h[T]}$, and $\frac{\partial l}{\partial h[t]}$ for all $t > T$. Please write down the expression for $\frac{\partial l}{\partial h[T]}$ for AttentionRNN(k).

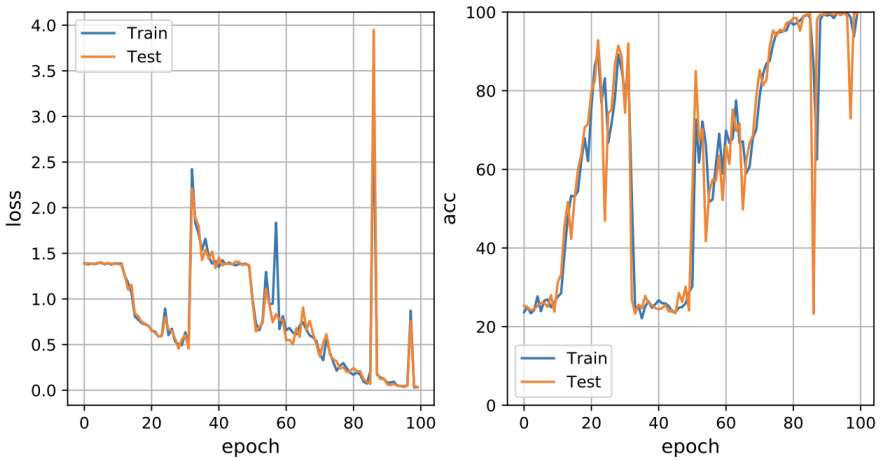
$\frac{\partial l}{\partial h[T]} = \sum_{t=T+1}^{\infty} \frac{\partial l}{\partial h[t]} \frac{\partial h[t]}{\partial h[T]}$

$\frac{\partial l}{\partial h[T]} = \sum_{t=T+1}^{\infty} \frac{\partial l}{\partial h[t]} \left(a_0[t] \frac{\partial v_0[t]}{\partial h[T]} + a_1[t] \frac{\partial v_1[t]}{\partial h[T]} \right)$

$\frac{\partial l}{\partial h[T]} = \sum_{t=T+1}^{T+2} \frac{\partial l}{\partial h[t]} a_0[t] \frac{\partial v_0[t]}{\partial h[T]} + \sum_{t=T+1}^{T+3} \frac{\partial l}{\partial h[t]} a_1[t] \frac{\partial v_1[t]}{\partial h[T]}$

1.3 Debugging loss curves

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. What caused the spikes on the left?

There are many reasons this can occur. The learning rate is too high such that the optimization algorithm does not find the local minimum. Another reason is that the gradients might be noisy.

2. How can they be higher than the initial value of the loss?

During training the model could encounter unrepresentative data, i.e., outliers. Overfitting would be another concern.

3. What are some ways to fix them?

Adjust learning rate (perhaps with a scheduluer) or implement gradient clipping. Could also use a different model, like an LSTM.

4. 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 parameters are randomly initialized.