ECE 684: Natural Language Processing

Mehak Arora

October 23, 2024

1 Question 1

The PyTorch LSTM operates with the following equations:

1. Input Gate:
$$i_t = \sigma(W_{ii}x_t + W_{hi}h_{t-1} + b_{ii} + b_{hi}) \tag{1}$$

2. Forget Gate:
$$f_t = \sigma(W_{if}x_t + W_{hf}h_{t-1} + b_{if} + b_{hf}) \tag{2}$$

3. Cell Gate:
$$g_t = \tanh(W_{ig}x_t + W_{hg}h_{t-1} + b_{ig} + b_{hg}) \tag{3}$$

4. Cell State:
$$c_t = f_t \circ c_{t-1} + i_t \circ g_t \tag{4}$$

5. Output Gate:
$$o_t = \sigma(W_{io}x_t + W_{ho}h_{t-1} + b_{io} + b_{ho}) \tag{5}$$

6. Hidden State:
$$h_t = o_t \circ \tanh(c_t) \tag{6}$$

Here, σ represents the sigmoid activation, tanh is the hyperbolic tangent, and $\circ$ is element-wise multiplication (hadamard product).

The RNN as defined by pytorch is (with sigmoid non-linearity) :

$$h_t = \sigma(x_t W_{ih}^T + b_{ih} + h_{t-1}W_{hh}^T + b_{hh}) \tag{7}$$

To see how equation (7) is a subset of the LSTM, we need to show that by setting the weights of matrices used in the LSTM equations (1-6) in some known manner, we are able to get the hidden state of the LSTM (defined by equation (6)) at every time t to be equal to the hidden state of the RNN (defined by equation (7)).

We can see that this reduces to making sure that c_t at every time point t is a known, reproducible quantity. That is, c_t is known and $c_t = c_{t-1}$

To achieve this, c_t cannot be a function of x_t or h_{t-1}. So,

$$W_{ii} = 0_{ii}, \quad W_{hi} = 0_{hi}$$
$$W_{if} = 0_{if}, \quad W_{hf} = 0_{hf}$$
$$W_{ig} = 0_{ig}, \quad W_{hg} = 0_{hg}$$

Where all matrices of the gates (except output gate) are initialized to zero.

For ease of computation, we can set all the bias matrices of the input and forget gates to zero. That is,

$$b_{ii} = 0_{ii}, \quad b_{hi} = 0_{hi}$$
$$b_{if} = 0_{if}, \quad b_{hf} = 0_{hf}$$

Since these two gates are sigmoid activated, and $\sigma(0) = 0.5$, we get

$$f_t = \mathbf{1} * 0.5, \quad i_t = \mathbf{1} * 0.5$$

Where $\mathbf{1}$ is the matrix of ones.
Thus equation (4) reduces to

$$c_t = \mathbf{1} * 0.5 \circ c_{t-1} + \mathbf{1} * 0.5 \circ g_t \tag{8}$$

To make sure c_t is known and deterministic, we want $c_t = c_{t-1}$. For this, $g_t = c_{t-1}$.

$$\begin{aligned}
&\implies g_t = \tanh(W_{ig}x_t + W_{hg}h_{t-1} + b_{ig} + b_{hg}) = c_{t-1} \\
&\implies W_{ig}x_t + W_{hg}h_{t-1} + b_{ig} + b_{hg} = \tanh^{-1}(c_{t-1}) \\
&\implies b_{ig} + b_{hg} = \tanh^{-1}(c_{t-1}) \qquad \text{Since we set the W matrices to zero} \qquad (9) \\
&\text{Let us set } b_{ig} = b_{hg} \\
&\implies b_{ig} = b_{hg} = 0.5 * \tanh^{-1}(c_{t-1})
\end{aligned}$$

If we choose c_0 to be some known number (with a *nice* value of tanh), like 0.5. Then we can set $b_{ig} = b_{hg} = 0.5 * \tanh^{-1}(c_{t-1}) = 0.5 * \tanh^{-1}(0.5)$.
Therefore,

$$\begin{aligned}
&c_t = \mathbf{1} * 0.5 \circ c_{t-1} + \mathbf{1} * 0.5 \circ \tanh(0.5 * \tanh^{-1}(c_{t-1}) + 0.5 * \tanh^{-1}(c_{t-1})) \\
&c_t = \mathbf{1} * 0.5 \circ c_{t-1} + \mathbf{1} * 0.5 \circ c_{t-1}) \qquad (10) \\
&c_t = c_{t-1} \text{ which we have chosen as } 0.5
\end{aligned}$$

Finally, we need,

$$\begin{aligned}
h_t = o_t \circ \tanh(c_t) &= \sigma(x_t W_{ih}^T + b_{ih} + h_{t-1} W_{hh}^T + b_{hh}) \\
\sigma(W_{io}x_t + W_{ho}h_{t-1} + b_{io} + b_{ho}) \circ \tanh(c_t) &= \sigma(x_t W_{ih}^T + b_{ih} + h_{t-1} W_{hh}^T + b_{hh})
\end{aligned} \tag{11}$$

Setting

$$b_{io} = b_{ih}, \quad b_{ho} = b_{hh}$$

$$W_{io} = W_{ih}, \quad W_{ho} = W_{hh}$$

We get that $h_t^{LSTM} = \frac{1}{\tanh(0.5)} h_t^{RNN}$.
So the RNN output and hidden state are a scaled version of that achieved by the LSTM. To get the two to exactly match, we can choose our c_t to be a large number (say 10000). This way the numerical value of $\tanh(c_t) = 1$ for standard floating point precision (implemented in numpy or pytorch).

2 Question 2

Let the input at any time t, x_t be a 4×1 one-hot encoded vector of the number of times the words/phrases "bad", "good", "not", and "uh" occur. Such that
$x_t(0)$ = number of occurrences of "bad"
$x_t(1)$ = number of occurrences of "good"
$x_t(2)$ = number of occurrences of "not"
$x_t(3)$ = number of occurrences of "uh"

Thus, the input size is 4×1.

Let us assume that the size of the hidden state is 2. The first dimension holds the "Pseudo" sentiment score (i.e., some scaled version of it), and the second dimension is an *almost* boolean type variable that holds information about whether the previous word was "not".

$x = [\text{bad good not uh }]^T$ and $h = [\text{scaled sentiment score previous word is "not"}]^T$

Since the range of g_t is [-1, 1], it is the gate that can encode information about whether the sentiment was positive or negative, i.e., whether the cumulative sentiment score should be *added to* or *subtracted from.*

Setting

$$W_{ig} = \begin{bmatrix} -100 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \end{bmatrix} \quad W_{hg} = \begin{bmatrix} 0 & -200 \\ 0 & 0 \end{bmatrix}$$

All biases will be set to zero.
Essentially, all occurrences of good will set $g_t = 1$ and vice versa for "bad". However, if the second element of the hidden state is **not zero**, then the sign of g_t will be reversed. Also, if the input is "not", then the second element will be set to 1, and will be 0 otherwise.

The input gate will then be used to scale the output of g_t to encode the information of whether the bad sentiment was from the word "bad", or from the phrase "not good."

Setting

$$W_{ii} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad W_{hi} = \begin{bmatrix} 0 & 100 \\ 0 & 0 \end{bmatrix}$$

$$b_{ii} = \begin{bmatrix} 0 \\ 100 \end{bmatrix} \quad b_{hi} = \begin{bmatrix} 0 \\ 100 \end{bmatrix}$$

This will ensure that if the previous element is not "not", then the sum inside the sigmoid function will always sum to 0, so the value of $i_t[0] = 0.5$. When the second element of h_t is set to greater than zero, then the value of $i_t[1] = 1$. The required scale of the sentiment score needs to be multiplied by two to reflect what is asked in the question, but for now we assume we can scale appropriately at the output.
The biases are set such that $i_t[1] = 1$ always.

Then, we do not want to forget $c_t[0]$, but we do want to wipe out $c_t[1]$, as this changes with every time step t. Thus,
Setting

$$W_{if} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad W_{hf} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$b_{if} = \begin{bmatrix} 100 \\ -100 \end{bmatrix} \quad b_{hf} = \begin{bmatrix} 100 \\ -100 \end{bmatrix}$$

Thus, $c_t[0]$ holds $0.5 \times$ sentiment score at every iteration t. This is **assuming** $h_t[1] \approx 1$. Or at least it is not too small. To ensure this, we want $h_t[1] = 1$ if $c_t[1] = 1$.

$h_t = o_t \circ \tanh(c_t)$.
But $\tanh(1) \approx 0.76$

So $\tanh(c_t)[0] \in [-1, 1]$ and $\tanh c_t[1] \in 0, 0.76$. All we need from o_t is to transmit exactly the value at the second element. thus,

$$W_{io} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad W_{hf} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$b_{if} = \begin{bmatrix} 100 \\ 100 \end{bmatrix} \quad b_{hf} = \begin{bmatrix} 100 \\ 100 \end{bmatrix}$$

Thus, as our only requirement for $h_t[1]$ was for it to be greater than 0, our model will still work. At every time step, the sentiment score $= 2 \times c_t[0]$.