Name: Yuxuan Zhang

NetID: yz885

Date: October 25, 2024

LSTM HW6

1. *Proof.* From the definition given by PyTorch, we know that the simple RNN with sigmoid nonlinearity updates its hidden state h_t using the following equation:

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

where σ is the sigmoid function. Also from PyTorch, we know that the LSTM is defined as follow:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t). \end{aligned}$$

Notice that o_t is the gate with sigmoid non-linearity. Hence, we can show that the simple RNN with sigmoid non-linearity is a subset of LSTM letting $\tanh(c_t) \approx 1$. We know that $\lim_{c_t \to \infty} \tanh(c_t) = 1$. This means that we need c_t to be very large.

- f_t Notice that since f_t uses the sigmoid activation function, $0 < f_t < 1$. f_t controls how much of the previous cell state c_{t-1} is retained. Then to ensure large increases in c_t, we need to set $f_t \to 1$.
- c_0 To ensure c_t diverges to infinity, we can set the initial state with large positive values.
- i_t Notice that i_t also involves the sigmoid activation function, which means that $0 < i_t < 1$. i_t controls how much of the new information is written to the cell state. Then we can set $i_t \to 1$ to ensure c_t goes to infinity.
- g_t Since g_t involve tanh, $-1 < g_t < 1$. Then we can set $g_t \to 1$ for c_t to approach to infinity.

Now we can rewrite the recurrence function as

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ &= f_t^t c_0 + t \cdot 1 \cdot 1 \\ &= f_t^t c_0 + t. \end{aligned}$$

Notice that $\lim_{t \to \infty} c_t = \infty$. Also, we mentioned that c_0 can be initialized with large values, then $\tanh(c_t) \to 1$. Then,

$$h_t \to o_t \odot 1 = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}),$$

which is the recurrence equation of a simple RNN with sigmoid nonlinearity. Hence, the simple RNN with sigmoid nonlinearity is a subset of the LSTM.

For the second part of the question, we can set all weights $W = 0$ with a large bias. Then,

$$\begin{aligned} f_t = i_t &= \sigma(0 + \text{large bias}) \to 1 \\ g_t &= \tanh(0 + \text{large bias}) \to 1. \end{aligned}$$

Then these conditions will ensure the LSTM to mimic the RNN by what we have shown above. $\square$

2. **One-hot Encoding**
According to the alphabetical order, we set

$$
\begin{aligned}
\text{"bad"} &= [1, 0, 0, 0] \\
\text{"good"} &= [0, 1, 0, 0] \\
\text{"not"} &= [0, 0, 1, 0] \\
\text{"uh"} &= [0, 0, 0, 1].
\end{aligned}
$$

General Workflow
Notice the recurrence relationship of c_t is

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t.$$

We want to delay the sentiment score by 1. That is, c_t contains the cumulated sentiment score at $t-1$, c_{t-1} contains the cumulated sentiment score at $t-2$, and g_t calculates the sentiment score at time $t-1$ by utilizing the hidden state h_{t-1}. Let's then consider each gate separately.

Gate Specification

Input i_t We want $i_t = 1_{1\times 6}$ for the following reason:
The equation we are given is

$$\sum \text{"good"} - \sum \text{"bad"} - 2\sum \text{"not good"} + 2\sum \text{"not bad"}.$$

This means that in each iteration, we are either adding 1, subtracting 1 or doing nothing (when the current word is "not" or "uh"). Thus, we need to completely keep the updated information, namely, $i_t = [1, 1, 1, 1, 1, 1]^T$. To do this, we just need to set

$$
\begin{aligned}
W_{ii} &= 0_{6\times 4} \\
W_{hi} &= 0_{6\times 6} \\
b_{ii} &= 0_{1\times 6} \\
b_{hi} &= 100_{1\times 6}.
\end{aligned}
$$

Then, we have

$$
\begin{aligned}
i_t &= \sigma(0 + b_{ii} + 0 + b_{hi}) \\
&= \sigma([100, 100, 100, 100, 100, 100]^T) \\
&\approx [1, 1, 1, 1, 1, 1]^T.
\end{aligned}
$$

Forget f_t We want $f_t = [0, 0, 0, 0, 0, 1]^T$ for the following reason:
In the forget gate, since we only care about the sentiment score, we only need to set the element corresponding to the sentiment score to 1. In our case, we set the last element to be the sentiment score. To achieve this, we just need to set

$$
\begin{aligned}
W_{if} &= 0_{6\times 4} \\
W_{hf} &= 0_{6\times 6} \\
b_{if} &= [-100, -100, -100, -100, -100, 100]^T \\
b_{hf} &= 0_{1\times 6}.
\end{aligned}
$$

Then, we have

$$
\begin{aligned}
f_t &= \sigma(0 + b_{if} + 0 + 0) \\
&= \sigma([-100, -100, -100, -100, -100, 100]^T) \\
&\approx [0, 0, 0, 0, 0, 1].
\end{aligned}
$$

Cell g_t We want $g_t = [1, 1, 1, 1, 1, \text{delayed increment score}]$.
Notice that

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}),$$

and we only want to use this gate to determine the sentiment score at $t-1$ through the hidden state h_{t-1}. To achieve this, since we do not need to handle x_t at this point, we can set $W_{ig} = 0_{6\times 4}$ and $b_{ig} = 0_{1\times 6}$. To determine the sentiment score, according to the equation, we have

$$\begin{aligned} \text{bad} &= -1 \\ \text{good} &= +1 \\ \text{not bad} &= 2 \cdot 1 - 1 = +1 \text{ (since "bad" alone will also be counted)} \\ \text{not good} &= -2 \cdot 1 + 1 = -1 \text{ (since "good" alone will also be counted).} \end{aligned}$$

Thus, we can design W_{hg} as follows:

$$W_{hg} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -100 & 100 & 0 & 100 & -100 & 0 \end{bmatrix}.$$

We design our hidden state matrix to be flags of the words, that is

$$h_{t-1} = [I_{\text{bad}}, I_{\text{good}}, I_{\text{not}}, I_{\text{not bad}}, I_{\text{not good}}, 0]^T.$$

Then,

$$(W_{hg}h_{t-1})_6 = -100I_{\text{bad}} + 100I_{\text{good}} + 100I_{\text{not bad}} - 100I_{\text{not good}},$$

which serves the function of determining sentiment scores. The first five elements of $W_{hg}h_{t-1}$ are zeros now. We then set

$$b_{ig} = [100, 100, 100, 100, 100, 0]^T.$$

Then,

$$\begin{aligned} g_t &= \tanh(0 + 0 + W_{hg}h_{t-1} + b_{hg}) \\ &= \tanh([100, 100, 100, 100, 100, (W_{hg}h_{t-1})_6]^T) \\ &= [1, 1, 1, 1, 1, \text{delayed increment score}] \end{aligned}$$

where delayed increment score $= \tanh((W_{hg}h_{t-1})_6) \in \{-1, 0, 1\}$.

Output o_t We want o_t to store flags of the current word given the delayed information stored in h_{t-1}. That is, we want

$$o_t = [I_{\text{bad}}, I_{\text{good}}, I_{\text{not}}, I_{\text{not bad}}, I_{\text{not good}}, 0].$$

To do so, we need $W_{io}x_t$ to handle the current word, and $W_{ho}h_{t-1}$ to determine if the previous word is "not". Then we set

$$W_{io} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$W_{ho} = \begin{bmatrix} 0 & 0 & -100 & 0 & 0 & 0 \\ 0 & 0 & -100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Then, $W_{io}x_t = [100I_{\text{bad}}, 100I_{\text{good}}, 100I_{\text{not}}, 100I_{\text{bad}}, 100I_{\text{good}}, 0]^T$, and $W_{ho}h_{t-1} = [-100I'_{\text{not}}, -100I'_{\text{not}}, 0, 100I'_{\text{not}}, 100I'_{\text{not}}, 0]^T$ where I'_{not} is the "not" flag at $t-1$. Then, we set

$$b_{ho} = [-50, -50, -50, -150, -150, -50]^T.$$

We set the bias this way because, as an example, if we encounter "not bad", we would need $o_t = [0, 0, 0, 1, 0, 0]^T$. Since $W_{io}x_t = [100, 0, 0, 100, 0, 0]^T$, and $W_{ho}h_{t-1} = [-100, -100, 0, 100, 100, 0]^T$, we have

$$W_{io}x + W_{ho}h_{t-1} = [0, -100, 0, 200, 100, 0]^T.$$

However, since $\sigma(0) = 0.5$, we need to add a large negative number to scale down these elements by an appropriate amount, so that only the correct element is activated by the sigmoid function.

Hidden h_t Lastly, we update the hidden state at last using the function $h_t = o_t \odot \tanh(c_t)$.

Result Validation

We use Python to validate if our configuration is correct

```python
### Author: Yuxuan Zhang
### Date: 10/24/2024
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def tanh(x):
    return np.tanh(x)

def lstm(sequence):
    # Define the one-hot encoding
    word_dict = {
        "bad": np.array([1, 0, 0, 0]),
        "good": np.array([0, 1, 0, 0]),
        "not": np.array([0, 0, 1, 0]),
        "uh": np.array([0, 0, 0, 1])
    }

    # Initialize h_0 and c_0
    h_t = np.zeros(6)
    c_t = np.zeros(6)

    # For simplicity, set i_t and f_t to constants
    i_t = np.ones(6)
    f_t = np.array([0, 0, 0, 0, 0, 1])

    # Cell Gate matrices
    W_ig = np.zeros((6, 4))
    W_hg = np.array([
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
```

```python
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [-100, 100, 0, 100, -100, 0]
    ])

    b_ig = np.array([100, 100, 100, 100, 100, 0])
    b_hg = np.zeros(6)

    # Output gate matrices
    W_io = np.array([
        [100, 0, 0, 0],
        [0, 100, 0, 0],
        [0, 0, 100, 0],
        [100, 0, 0, 0],
        [0, 100, 0, 0],
        [0, 0, 0, 0]
    ])
    W_ho = np.array([
        [0, 0, -100, 0, 0, 0],
        [0, 0, -100, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 100, 0, 0, 0],
        [0, 0, 100, 0, 0, 0],
        [0, 0, 0, 0, 0, 0]
    ])

    b_io = np.zeros(6)
    b_ho = np.array([-50, -50, -50, -150, -150, -50] )

    for word in sequence:
        print(f"Processing word: {word}")

        x_t = word_dict[word]
        g_t = tanh(np.dot(W_ig, x_t) + b_ig + np.dot(W_hg, h_t) + b_hg)
        c_t = f_t * c_t + i_t * g_t
        o_t = sigmoid(np.dot(W_io, x_t) + b_io + np.dot(W_ho, h_t) + b_ho)
        h_t = o_t * tanh(c_t)

        print("Updated delayed total sentiment score:", c_t[-1])
        print("-" * 50)

def main():
    sequence = ["uh", "good", "good", "not", "not", "bad", "bad", "uh"]
    lstm(sequence)

if __name__ == "__main__":
    main()
```

```
d:/NLP-Course-Project/LSTM/LSTM.py
Processing word: uh
Updated delayed total sentiment score: 0.0
--------------------------------------------------
Processing word: good
Updated delayed total sentiment score: 0.0
--------------------------------------------------
Processing word: good
Updated delayed total sentiment score: 1.0
```

```
--------------------------------------------------
Processing word: not
Updated delayed total sentiment score: 2.0
--------------------------------------------------
Processing word: not
Updated delayed total sentiment score: 2.0
--------------------------------------------------
Processing word: bad
Updated delayed total sentiment score: 2.0
--------------------------------------------------
Processing word: bad
Updated delayed total sentiment score: 3.0
--------------------------------------------------
Processing word: uh
Updated delayed total sentiment score: 2.0
--------------------------------------------------
```

Notice that the sentence that we used is "uh good good not not bad bad uh". There are 2 "good"s, 2 "bad"s, and 1 "not bad", using the equation, we know the total sentiment score should be

$$1 \cdot 2 - 1 \cdot 2 + 1 \cdot 2 = 2.$$

This matches with what we get from the code, which validates our configurations above.