



# تمرین چهارم

## NLP

یاسمین مدنی  
۹۷۵۳۲۲۶۵

---

## فهرست

۲	..... سوال تئوری
۳	..... :Backpropagation through time (BPTT) in RNNs
۶	..... LSTM
۱۰	..... The error gradients in an LSTM network
۱۲	..... Preventing the error gradients from vanishing
۱۵	..... سوال عملی
۱۵	..... سوال ۱
۱۷	..... سوال ۲

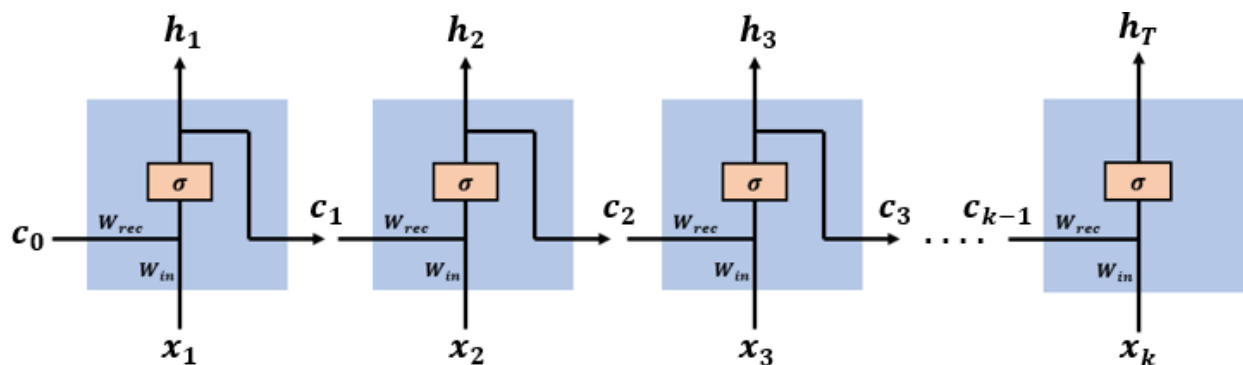
## سوال تئوری

RNN ها به مدل ها کمک می کنند تا تسک ها را وابسته به زمان و از روی داده های متوالی انجام دهند. مانند پیش بینی بازار سهام، ترجمه ماشینی، تولید متن و ...

با این حال، RNN ها با مشکل ناپدید شدن گرادیان روبرو هستند، که یادگیری توالی داده های طولانی را با مشکل مواجه می کند.

گرادیان ها اطلاعات مورد استفاده در به روزرسانی پارامتر RNN را حمل می کنند و وقتی گرادیان کوچک تر و کوچک تر می شود، به روزرسانی ها ناچیز می شوند که به این معنی است که هیچ یادگیری واقعی انجام نمی شود.

برای مثال ما با یک RNN با یک لایه پنهان ساده با یک توالی خروجی کار خواهیم کرد. این شبکه به شکل زیر است:



شبکه دارای یک دنباله ورودی از بردارها  $[x(1), x(2), \dots, x(k)]$  است، در مرحله زمانی  $t$  شبکه دارای یک بردار ورودی  $x(t)$  است. اطلاعات گذشته و دانش آموخته شده در بردارهای وضعیت شبکه  $[c(1), c(2), \dots, c(k-1)]$  کدگذاری می شود، در مرحله زمانی  $t$  شبکه دارای یک بردار حالت ورودی  $c(t-1)$  است. بردار ورودی  $x(t)$  و بردار حالت  $c(t-1)$  به هم پیوسته اند تا بردار ورودی کامل را در مرحله زمانی  $t$  تشکیل دهند.

شبکه دو ماتریس وزن دارد:  $W_{rec}$  و  $W_{in}$  که  $c(t-1)$  و  $x(t)$  را به هم متصل می کند.

برای سادگی، بردارهای بایاس را در محاسبات خود حذف می کنیم و  $W = [W_{rec}, W_{in}]$  را نشان می دهیم.

تابع sigmoid به عنوان تابع فعال سازی در لایه پنهان استفاده می شود. شبکه یک بردار واحد را در آخرین مرحله زمانی خروجی می دهد (RNN) ها می توانند در هر مرحله زمانی یک بردار تولید کنند، اما ما از این مدل ساده تر استفاده خواهیم کرد).

### :Backpropagation through time (BPTT) in RNNs

پس از آنکه RNN بردار پیش بینی  $h(k)$  را خروجی می دهد، خطای پیش بینی  $E(k)$  را محاسبه کرده و از الگوریتم Back Propagation Through Time برای محاسبه گرادیان استفاده می کنیم.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W}$$

گرادیان برای به روز رسانی پارامترهای مدل استفاده می شود

$$W \leftarrow W - \alpha \frac{\partial E}{\partial W}$$

و فرآیند یادگیری را با استفاده از الگوریتم Gradient Descent (GD) ادامه می دهیم (در این کار از نسخه اصلی GD استفاده می کنیم).

فرض کنید ما یک فرایند یادگیری داریم که شامل مراحل زمانی  $T$  است، گرادیان خطا در گام زمانی  $k$  به صورت زیر داده می شود:

$$\begin{aligned}\frac{\partial E_k}{\partial W} &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \dots \frac{\partial c_2}{\partial c_1} \frac{\partial c_1}{\partial W} \\ &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left( \prod_{t=2}^k \frac{\partial c_t}{\partial c_{t-1}} \right) \frac{\partial c_1}{\partial W} \quad (1)\end{aligned}$$

توجه داشته باشید که از آنجایی که  $W=[W_{rec}, W_{in}]$ ،  $c(t)$  را می توان به صورت زیر نوشت:

$$c_t = \sigma (W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t)$$

مشتق  $c(t)$  را محاسبه می کنیم و داریم:

$$\begin{aligned}\frac{\partial c_t}{\partial c_{t-1}} &= \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot \frac{\partial}{\partial c_{t-1}} [W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t] \\ &= \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \quad (2)\end{aligned}$$

با ترکیب روابط ۱ و ۲ داریم:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left( \prod_{t=2}^k \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \right) \frac{\partial c_1}{\partial W}$$

آخرین عبارت زمانی که  $k$  بزرگ است ناپدید می شود، این به دلیل مشتق تابع فعال سازی  $\tanh$  است که کوچکتر از ۱ است.

اگر وزن های  $W_{rec}$  به اندازه کافی بزرگ باشند که بر مشتق کوچکتر  $\tanh$  غلبه کنند، حاصل ضرب مشتقات می تواند بسیار بزرگ شود این به عنوان مشکل انفجار گرادیان شناخته می شود. داریم:

$$\prod_{t=2}^k \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \rightarrow 0$$

پس در یک زمان استپ زمانی  $k$  داریم:

$$\frac{\partial E_k}{\partial W} \rightarrow 0$$

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W} \rightarrow 0$$

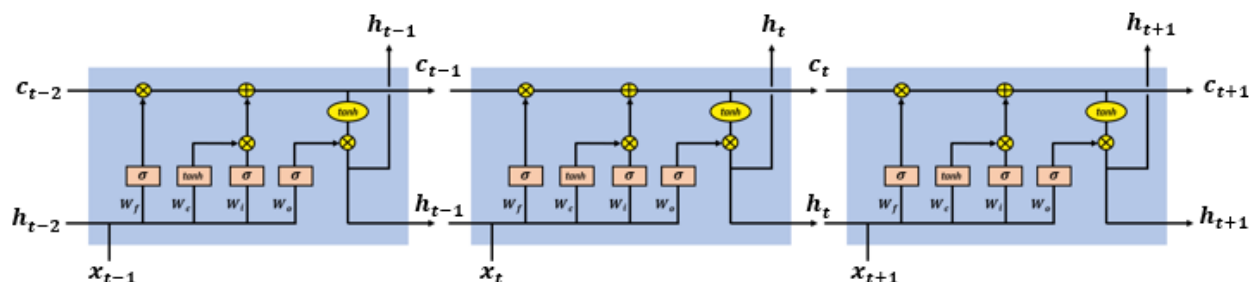
به روز رسانی وزن شبکه :

$$W \leftarrow W - \alpha \frac{\partial E}{\partial W} \approx W$$

و هیچ یادگیری قابل توجهی در زمان معقول انجام نخواهد شد.

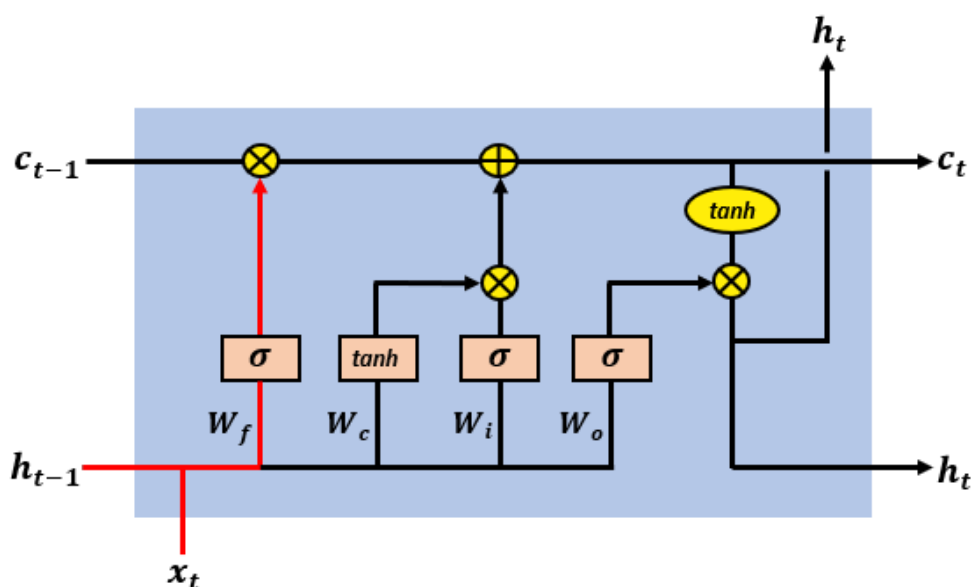
## LSTM

یک شبکه LSTM دارای یک بردار ورودی  $[x(t), h(t-1)]$  در مرحله زمانی  $t$  است. وضعیت سلول شبکه با  $c(t)$  نشان داده می شود. بردارهای خروجی عبور داده شده از شبکه بین مراحل زمانی متوالی  $t, t+1$  با  $h(t)$  نشان داده می شوند.



یک شبکه LSTM دارای سه گیت است که حالت های سلولی را به روز می کنند و کنترل می کنند، این ها گیت فراموشی، گیت ورودی و گیت خروجی هستند. گیت ها از توابع فعال سازی تانژانت هذلولی و سیگموئید استفاده می کنند.

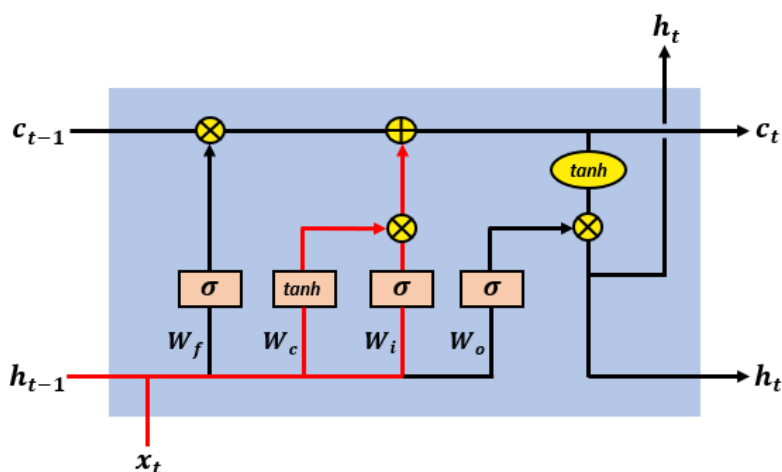
گیت فراموشی کنترل می کند که، با توجه به اطلاعات جدید وارد شده به شبکه چه اطلاعاتی در حالت سلول باید فراموش شود.



خروجی گیت فراموشی:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t])$$

گیت ورودی کنترل می کند که با توجه به اطلاعات ورودی جدید، چه اطلاعات جدیدی در حالت سلول کدگذاری می شود.



خروجی گیت ورودی به شکل زیر است:

$$\tanh(W_c \cdot [h_{t-1}, x_t]) \otimes \sigma(W_i \cdot [h_{t-1}, x_t])$$

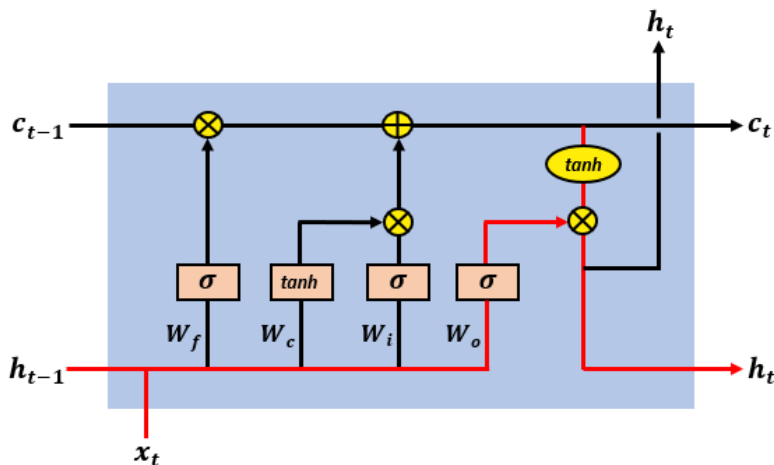
و برابر است با حاصلضرب عنصری خروجی دو لایه کاملاً متصل.

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t])$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$$



گیت خروجی کنترل می کند که چه اطلاعاتی که در حالت سلول کدگذاری شده است به عنوان ورودی در مرحله زمانی زیر به شبکه ارسال می شود، این کار از طریق بردار خروجی  $h(t)$  انجام می شود.



فعال سازی های گیت خروجی :

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t])$$

بردار خروجی سلول :

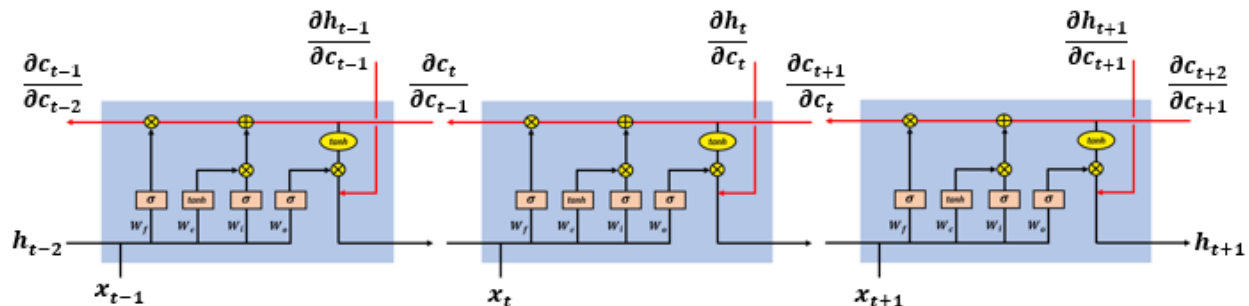
$$h_t = o_t \otimes \tanh(c_t)$$

وابستگی ها و روابط طولانی مدت در بردارهای حالت سلولی کدگذاری می شوند و این مشتق حالت سلولی است که می تواند از ناپدید شدن گرادیان های LSTM جلوگیری کند.

$$c_t = c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t$$

همانند مدل RNN، شبکه LSTM ما یک بردار پیش بینی  $h(k)$  را در گام زمانی  $k$ -ام خروجی می دهد. دانش کدگذاری شده در بردارهای حالت  $c(t)$  وابستگی ها و روابط بلندمدت در داده های متوالی را نشان می دهد. طول توالی داده ها می تواند صدها و حتی هزاران گام زمانی باشد که

یادگیری با استفاده از یک RNN پایه را بسیار دشوار می کند. ما گرادیان مورد استفاده برای به روز رسانی پارامترهای شبکه را محاسبه می کنیم، محاسبه در مراحل زمانی  $T$  انجام می شود.



همانطور که در RNN ها، عبارت خطای گرادیان با رابطه زیر بدست می آید.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W} \quad (3)$$

برای ناپدید شدن کامل گرادیان خطا، همه این زیر گرادیان ها باید ناپدید شوند. اگر ما (۳) را به عنوان یک سری از توابع در نظر بگیریم، بنا به تعریف، اگر دنباله مجموع جزئی آن به صفر گرایش پیدا کند، این سری به صفر همگرا می شود.

$$\sum_{t=1}^T \frac{\partial E_t}{\partial W} \rightarrow 0$$

$$S_n = \sum_{t=1}^n \frac{\partial E_t}{\partial W}$$

بنابراین اگر می خواهیم (۳) ناپدید نشود، شبکه ما باید احتمال ناپدید نشدن حداقل برخی از این گرادیان های فرعی را افزایش دهد، به عبارت دیگر، باعث شود که سری های گرادیان های فرعی در (۳) به صفر همگرا نشوند.

## The error gradients in an LSTM network

گرادیان خطا برای  $k$  استپ به شکل زیر است:

$$\begin{aligned} \frac{\partial E_k}{\partial W} &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \dots \frac{\partial c_2}{\partial c_1} \frac{\partial c_1}{\partial W} \\ &= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left( \prod_{t=2}^k \frac{\partial c_t}{\partial c_{t-1}} \right) \frac{\partial c_1}{\partial W} \quad (4) \end{aligned}$$

همانطور که دیدیم، عبارت زیر باعث از بین رفتن گرادیان ها می شود:

$$\prod_{t=2}^k \frac{\partial c_t}{\partial c_{t-1}}$$

در یک LSTM، بردار حالت  $c(t)$ ، به شکل زیر است:

$$\begin{aligned} c_t &= c_{t-1} \otimes \sigma(W_f \cdot [h_{t-1}, x_t]) \oplus \\ &\quad \tanh(W_c \cdot [h_{t-1}, x_t]) \otimes \sigma(W_i \cdot [h_{t-1}, x_t]) \end{aligned}$$

که می توان به صورت فشرده نوشت

$$c_t = c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t \quad (5)$$

توجه داریم که بردار حالت  $c(t)$  تابعی از عناصر زیر است که باید هنگام محاسبه مشتق در حین backpropagation در نظر گرفته شود:

$$c_{t-1}, f_t, \tilde{c}_t, i_t$$

مشتق (۵) :

$$\begin{aligned}
 \frac{\partial c_t}{\partial c_{t-1}} &= \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t] \\
 &= \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}} [\tilde{c}_t \otimes i_t] \\
 &= \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \frac{\partial c_{t-1}}{\partial c_{t-1}} \cdot f_t + \frac{\partial i_t}{\partial c_{t-1}} \cdot \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} \cdot i_t
 \end{aligned}$$

چهار عبارت مشتق را محاسبه می کنیم:

$$\begin{aligned}
 \frac{\partial c_t}{\partial c_{t-1}} &= \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot c_{t-1} \\
 &+ f_t \\
 &+ \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot \tilde{c}_t \\
 &+ \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot i_t
 \end{aligned}$$

چهار عنصر را که مشتق حالت سلولی هستند به این صورت مشخص می شوند:

$$A_t = \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot c_{t-1}$$

$$B_t = f_t$$

$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes \tanh'(c_{t-1}) \cdot i_t$$

additive گرادیان را به صورت می نویسیم:

$$\frac{\partial c_t}{\partial c_{t-1}} = A_t + B_t + C_t + D_t \quad (6)$$

با ترکیب (۶) و (۴) گرادیان حالت های LSTM به دست می آید:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left( \prod_{t=2}^k [A_t + B_t + C_t + D_t] \right) \frac{\partial c_1}{\partial W}$$

### Preventing the error gradients from vanishing

توجه داشته باشید که گرادیان شامل بردار فعال سازی های گیت فراموشی است که به شبکه اجازه می دهد تا با استفاده از به روزرسانی های پارامتر مناسب گیت فراموشی، مقادیر گرادیان را در هر مرحله زمانی بهتر کنترل کند. وجود فعال سازی های گیت فراموشی به LSTM این امکان را می دهد که در هر مرحله زمانی تصمیم بگیرد که اطلاعات خاصی فراموش نشود و پارامترهای مدل را بر این اساس به روزرسانی کند.

بیا بید مرور کنیم که این ویژگی چگونه به ما کمک می کند. برای مرحله  $k < T$ ، داریم که:

$$\sum_{t=1}^k \frac{\partial E_t}{\partial W} \rightarrow 0$$

سپس برای اینکه گرادیان ناپدید نشود، می توانیم یک به روز رسانی پارامتر مناسب گیت فراموشی را در مرحله زمانی  $k+1$  پیدا کنیم به طوری که

$$\frac{\partial E_{k+1}}{\partial W} \nrightarrow 0$$

وجود بردار فعال سازی دروازه فراموشی در ترم گرادیان همراه با ساختار additive است که به LSTM اجازه می دهد چنین به روز رسانی پارامتری را در هر مرحله زمانی پیدا کند و این نتیجه را به دست می آورد.

$$\sum_{t=1}^{k+1} \frac{\partial E_t}{\partial W} \nrightarrow 0$$

و گرادیان ناپدید نمی شود.

ویژگی مهم دیگری که باید به آن توجه کرد این است که گرادیان حالت سلولی یک تابع افزایشی است که از چهار عنصر  $A(t)$ ,  $B(t)$ ,  $C(t)$ ,  $D(t)$  تشکیل شده است. این ویژگی additive تعادل بهتر مقادیر گرادیان را در حین backpropagation امکان پذیر می کند. LSTM مقادیر چهار مؤلفه را به روزرسانی و متعادل می کند و احتمال ناپدید نشدن عبارت افزودنی را افزایش می دهد. به عنوان مثال، بگوییم که برای هر  $t$  در  $\{k, \dots, 2, 3\}$  چهار همسایگی مقادیر زیر را به عنوان یک ترکیب متعادل کننده در گرادیان خود در نظر می گیریم:

$$A_t \approx \overrightarrow{0.1}, B_t = f_t \approx \overrightarrow{0.7}, C_t \approx \overrightarrow{0.1}, D_t \approx \overrightarrow{0.1}$$

$$\begin{aligned} \prod_{t=2}^k [A_t + B_t + C_t + D_t] &\approx \prod_{t=2}^k [\overrightarrow{0.1} + \overrightarrow{0.7} + \overrightarrow{0.1} + \overrightarrow{0.1}] \\ &\approx \prod_{t=2}^k \overrightarrow{1} \nrightarrow 0 \end{aligned}$$

همانطور که ذکر شد، اگر مجموع رابطه (۳) از عباراتی با رفتار مشابه که همه به طور قابل توجهی بزرگتر از ۱ هستند، تشکیل شود، گرادیان های RNN نیز می توانند منفجر شوند.

LSTM ها با استفاده از یک ساختار گرادیان افزودنی منحصر به فرد که شامل دسترسی مستقیم به فعال سازی های گیت فراموشی است، مشکل را حل می کند و شبکه را قادر می سازد تا رفتار

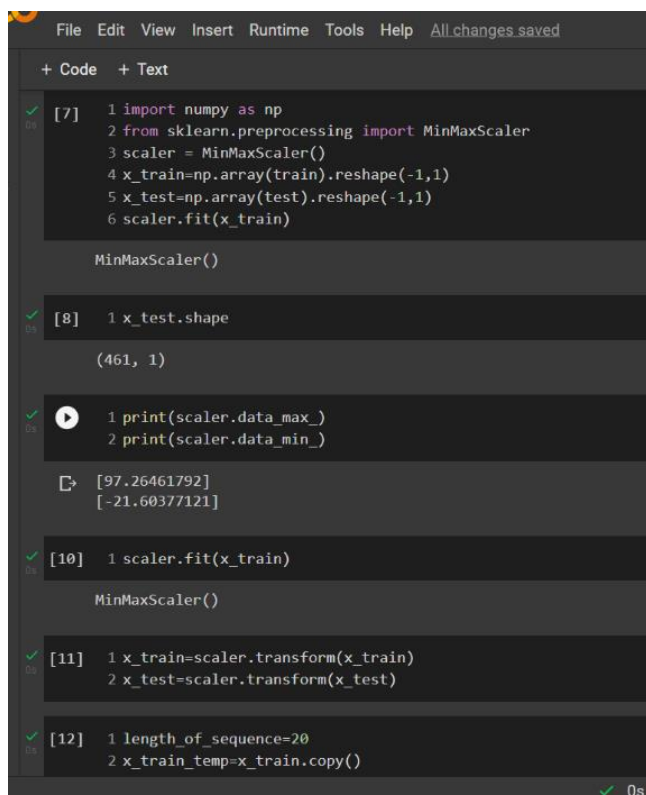
دلخواه را از گرادیان خطا با استفاده از به روز رسانی مکرر گیت ها در هر مرحله زمانی از فرآیند یادگیری به دست آورد.

## سوال عملی

### سوال ۱

برای حل این سوال یک مدل از چند لایه simple RNN ساخته و با استفاده از لایه drop out کمک می کنیم تا دقت برای داده های کمی که داریم افزایش داشته باشد.

```
6
7 def build_model():
8     model= tf.keras.Sequential([
9         layers.Input(shape=(20,1)),
10        tf.keras.layers.SimpleRNN(units=50,return_sequences=True),
11        tf.keras.layers.Dropout(rate=0.2),
12        tf.keras.layers.SimpleRNN(units=50,return_sequences=True),
13        tf.keras.layers.Dropout(rate=0.2),
14        tf.keras.layers.SimpleRNN(units=50,return_sequences=True),
15        tf.keras.layers.Dropout(rate=0.2),
16        tf.keras.layers.SimpleRNN(units=50,return_sequences=False),
17        tf.keras.layers.Dense(1, activation='relu')
18    ])
19
20     return model
```



```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

[7] 1 import numpy as np
    2 from sklearn.preprocessing import MinMaxScaler
    3 scaler = MinMaxScaler()
    4 x_train=np.array(train).reshape(-1,1)
    5 x_test=np.array(test).reshape(-1,1)
    6 scaler.fit(x_train)

MinMaxScaler()

[8] 1 x_test.shape
    (461, 1)

[9] 1 print(scaler.data_max_)
    2 print(scaler.data_min_)
    [97.26461792]
    [-21.60377121]

[10] 1 scaler.fit(x_train)
    MinMaxScaler()

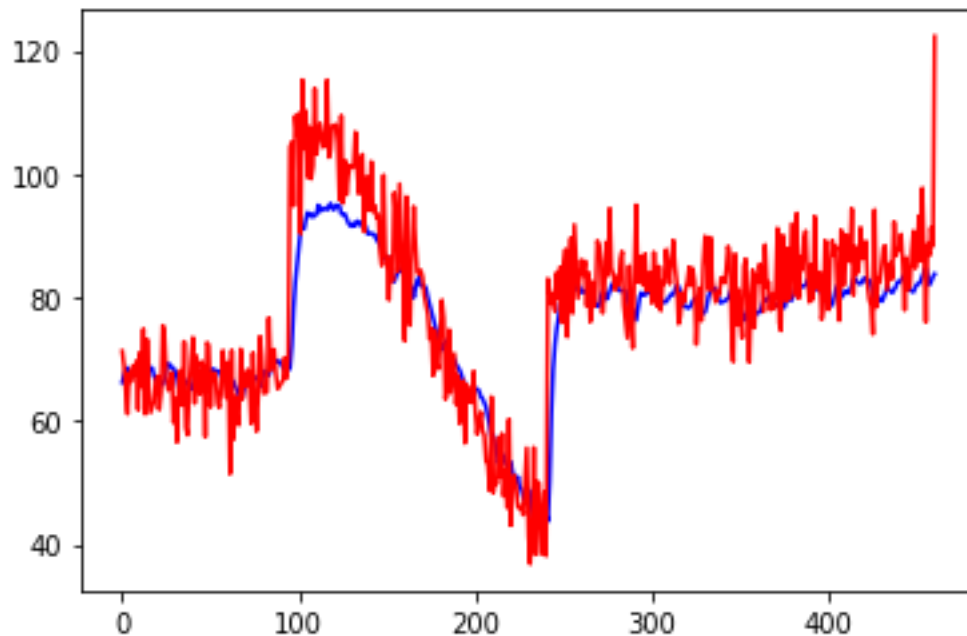
[11] 1 x_train=scaler.transform(x_train)
    2 x_test=scaler.transform(x_test)

[12] 1 length_of_sequence=20
    2 x_train_temp=x_train.copy()
```

همچنین نیاز است تا پیش از استفاده از داده ها آنها را اسکیل کنیم که این کار به راحتی با تابع MINMAXSCALER صورت می گیرد.



پس از انجام عملیات آموزش و تست داده ها را در نمودار زیر نمایش داده ایم.



نمودار قرمز داده های اصلی و نمودار آبی تخمین های مدل می باشد.

علاوه بر این مقدار `mean_absolute_error` خواسته شده در داک را نیز محاسبه می کنیم.

```
1 mean_absolute_error(x_test_to_show_vals, test_preds_to_show)
```

```
5.73820122435582
```

## سوال ۲

بخش های خواسته شده در داک سوال در نوت بوک کامل شده اند و دو تصویر زیر خروجی حالت های اول یعنی شبکه بدون attention و حالت دوم با استفاده از attention است.

```
1 TEST_SENTENCE = 'the air conditioning is working'
2 translated = translate_sentence(TEST_SENTENCE, rnn_encoder, rnn_decoder, None, args)
3 print("source:\t\t{} \ntranslated:\t{}".format(TEST_SENTENCE, translated))
```

```
source:      the air conditioning is working
translated:  ethay ainway onditionationway isway-ybay oullway
```

```
1 TEST_SENTENCE = 'the air conditioning is working'
2 translated = translate_sentence(TEST_SENTENCE, rnn_attn_encoder, rnn_attn_decoder, None, args)
3 print("source:\t\t{} \ntranslated:\t{}".format(TEST_SENTENCE, translated))
```

```
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:1944: UserWarning: nn.functional.sigmoid is deprecated. Use torch.sigmoid instead.
warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid instead.")
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:1933: UserWarning: nn.functional.tanh is deprecated. Use torch.tanh instead.
warnings.warn("nn.functional.tanh is deprecated. Use torch.tanh instead.")
source:      the air conditioning is working
translated:  ethay airway onindicay-inindgcay isway orkingway
```

به طور کلی در حالت نخست مشاهده میشود که تعداد بیشتری از کلمات به صورت نا درست ترجمه شده اند برای مثال **conditioning** و .. و این حالت تنها برای کلمات اولیه و کوتاه متن پاسخ نسبتا مناسبی خواهد داشت. و پس از مدتی اکسپوژر بایاس و عدم ارتباط با کاراکتر های پیشین کلمه ،ساخت کلمات بی معنی را نتیجه می دهد.

اما در حالت بعد و با استفاده از اتنشن تعداد کلمات بیشتری نسبت به قبل درست ترجمه می شوند و کلمات طولانی تر نیز بهتر ترجمه شده اند.اما در کلمات کوتاه تر مانند is به دلیل داشتن حافظه از کلمات قبلی مقداری دشوار تر عمل کرده است.