Computer Engineering Department

Natural Language Processing

Assignment 4

Ali Sedaghi
97521378

Table of contents

Theoretical questions

Q1- LSTM and Gradient Vanishing / Explosion[1]

برای پاسخ به این سوال ابتدا معادلات Simple RNN را بررسی کنیم تا متوجه شویم چرا این واحد ساده از انفجار یا ناپدید شدن گرادیان رنج می‌برد. معادلات Backprop این لایه به صورت زیر است:

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

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

اما این کار در T مرحله زمانی صورت می‌گیرد. گرادیان خطا در مرحله زمانی k به صورت زیر محاسبه می‌شود:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k} \cdots \frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W}$$

$$= \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k} \frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W} \qquad (1)$$

همچنین W را می‌توان به صورت Ct نوشت:

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

مشتق Ct نیز به صورت زیر محاسبه می‌شود:

$$\frac{\partial C_t}{\partial C_{t-1}} = \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot \frac{\partial}{\partial C_{t-1}}[W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t]$$

$$= \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \qquad (2)$$

[1] https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

با ترکیب روابط 1 و 2 رابطه گرادیان به صورت زیر به دست می‌آید:

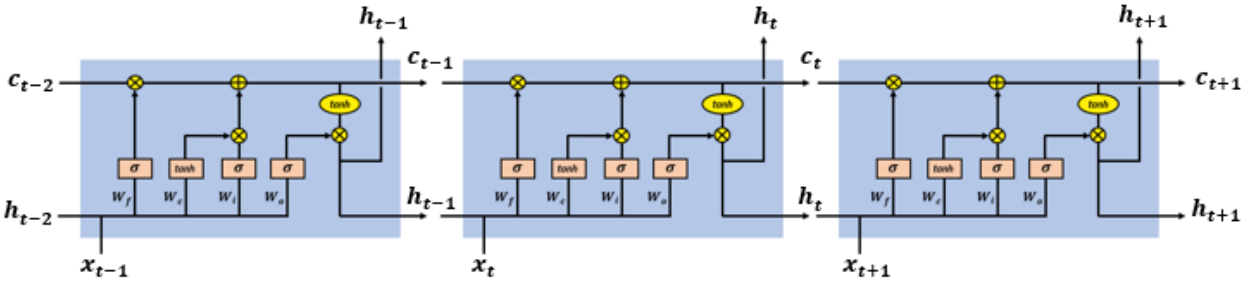
$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial C_k}\left(\prod_{t=2}^{k}\sigma'(W_{rec}\cdot c_{t-1} + W_{in}\cdot x_t)\cdot W_{rec}\right)\frac{\partial C_1}{\partial W}$$

هنگامی که k عددی بزرگ می‌شود (مراحل آخر زمانی) دو اتفاق ممکن است رخ دهد:

1- اگر مقدار مشتق تابع tanh عددی کوچکتر از 1 باشد: گرادیان به صفر میل پیدا می‌کند (Vanish) در نتیجه W تغییر نمی‌کند و شبکه Freeze می‌شود.

2- اگر بردار وزن Wrec به اندازه کافی بزرگ باشد تاثیر خود را روی مشتق تابع tanh می‌گذارد و مقدار گرادیان زیاد می‌شود (Explode)

حال بررسی می‌کنیم LSTM چگونه این مشکلات را حل می‌کند. نمای این ماژول در استپ‌های زمانی T-1 و T و T+1 در شکل زیر آورده شده است:



همچنین معادلات Forward این ماژول به صورت زیر است:

گیت Forget که کنترل می‌کند هنگامی که اطلاعات جدید وارد شبکه شد چه اطلاعاتی فراموش شوند:

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

گیت Input که مسئول کنترل Encoding اطلاعات جدید وارد شده به شبکه است:

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

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

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

گیت Output که مسئول این است که چه اطلاعات Encode شده ای به عنوان ورودی مرحله بعدی زمانی ارسال شوند:

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

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

با ترکیب موارد بالا به Cell State یک واحد LSTM می‌رسیم:

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

در واقع معادلات مشتق همین رابطه است که از محو شدگی یا انفجار گرادیان جلوگیری می‌کند.

گرادیان خطا در استپ زمانی k در ماژول LSTM به صورت زیر محاسبه می‌شود:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k} \dots \frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W}$$

$$= \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W} \qquad (4)$$

اما می‌دانیم ترم Ct در LSTM به صورت زیر است:

$$c_t = c_{t-1} \otimes \sigma(W_f \cdot [h_{t-1}, x_t]) \oplus$$

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

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

پس گرادیان آن به صورت زیر نوشته می‌شود:

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

$$= \frac{\partial}{\partial c_{t-1}}[c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}}[\tilde{c}_t \otimes i_t]$$

$$= \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \frac{\partial c_{t-1}}{\partial c_{t-1}} \cdot f_t + \frac{\partial i_t}{\partial c_{t-1}} \cdot \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} \cdot i_t$$

گرادیان بالا را می‌توان تبدیل به چندین جمع کرد:

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

با بازنویسی ترم‌های بالا تحت عناوین A و B و C و D داریم:

$$A_t = \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$
$$B_t = f_t$$
$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$
$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

ترم‌های بالا را در گرادیان Ct قرار می‌دهیم:

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

گرادیان نهایی به صورت زیر قابل محاسبه است:

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

حال برای جلوگیری از صفر شدن یا انفجار این گرادیان در استپ زمانی K کافیست Forget مناسبی را پیدا کنیم که باعث آن اتفاق نشود:

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

برای مثال اگر در یک شبکه مقادیر زیر برای A تا D انتخاب شوند گرادیان هیچگاه محو یا منفجر نمی‌شود:

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

زیرا:

$$\prod_{t=2}^{k}[A_t + B_t + C_t + D_t] \approx \prod_{t=2}^{k}[\overrightarrow{0.1} + \overrightarrow{0.7} + \overrightarrow{0.1} + \overrightarrow{0.1}]$$

$$\approx \prod_{t=2}^{k} \vec{1} \nrightarrow 0$$

در واقع این خاصیت منحصر به فرد Additive Gradient باعث می‌شود گیت Forget بتواند رفتار مطلوب را وارد گرادیان کند و از محوشدگی و انفجار گرادیان جلوگیری کند.

Practical questions

Q1- Simple RNN in Time Series

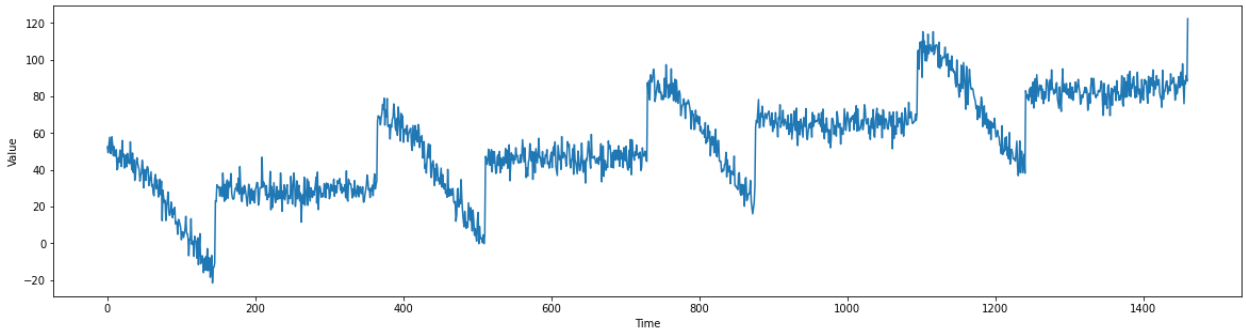
ابتدا دیتاست را بارگذاری می‌کنیم و آن را از فایل CSV به آرایه نامپای تبدیل می‌کنیم. شکل دو آرایه آموزش و ارزیابی به صورت زیر است:

Train Data Shape: (1000, 1)

Test Data Shape: (461, 1)

که رنک اول مربوط به زمان و رنک دوم مربوط به مقدار است.

نمودار سری زمانی این دیتاست (شامل آموزش و ارزیابی) را رسم می‌کنیم.



با استفاده از توابع create_train_set و create_test_set دیتاست را به پنجره‌های 20 خانه‌ای تبدیل می‌کنیم تا با داشتن 20 مقدار مقدار 21م را به دست آوریم.

شکل داده آموزشی به صورت زیر است:

x_train shape: (980, 20, 1)

y_train_shape: (980, 1)

در واقع در x دارای 980 ردیف هستیم که هر ردیف دارای 20 خانه است و هر خانه دارای یک مقدار است. در y نیز به ازای هر مجموعه 20 تایی از خانه‌ها مقدار خانه 21م را داریم.

شکل داده ارزیابی نیز به صورت زیر است:

x_train shape: (461, 20, 1)

y_train_shape: (461, 1)

که این اعداد مشابه قسمت آموزش هستند.

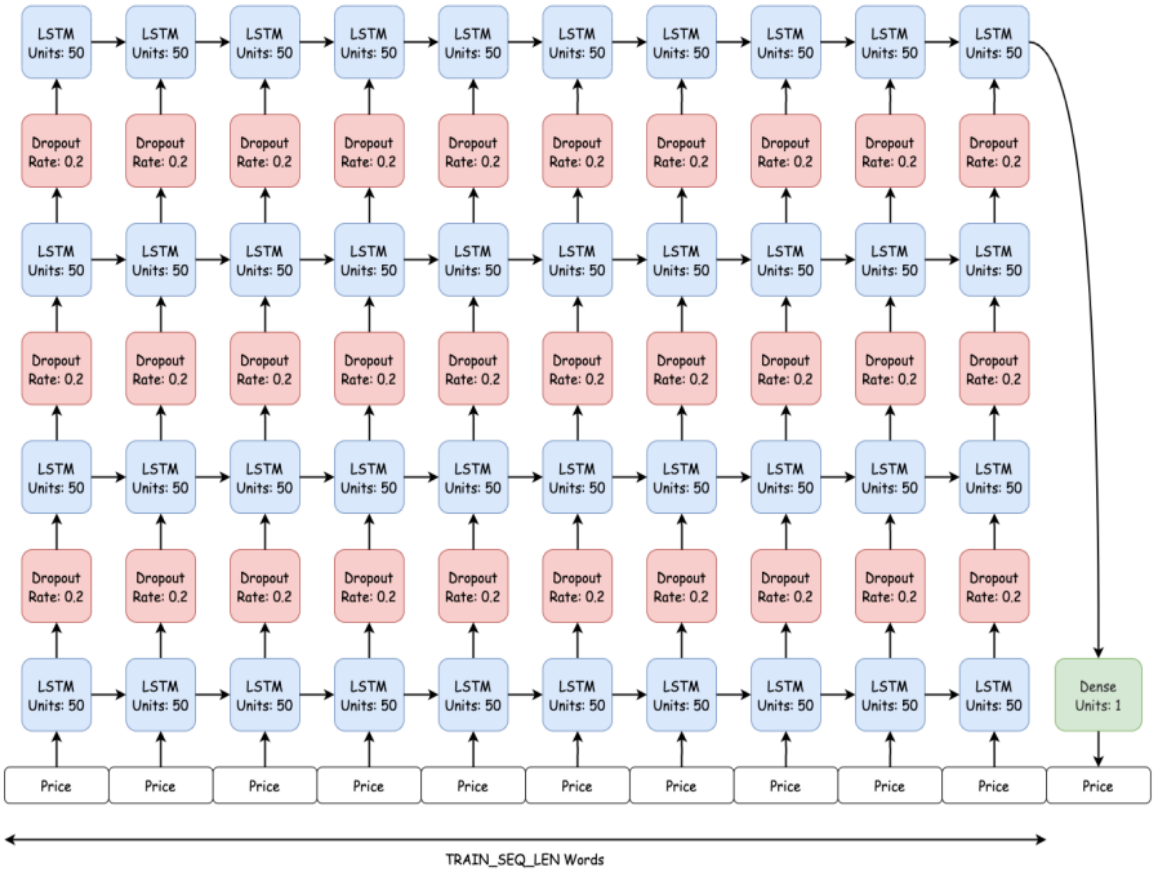
نکته: در صورت سوال گفته شده برای 440 روز ممکن پیش‌بینی کنید اما ما با استفاده از 20 روز آخر مجموعه آموزش به عنوان 20 روز اولیه داده ارزیابی توانستیم به ازای هر 461 روز پیش‌بینی داشته باشیم.

از مقیاس کننده Min Max که درون کتاب‌خانه SK Learn وجود دارد برای مقیاس کردن مقادیر دیتاست استفاده کردیم. رابطه این مقیاس کننده به صورت زیر است:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

این مقایس کننده بر روی داده آموزشی Fit شده و روی داده آموزش و ارزیابی اعمال شده است. در فاز اینفرنس نیز از معکوس (Inverse) این مقیاس کننده برای به دست آوردن مقادیر واقعی استفاده شده است.

از یک مدل بازگشتی (RNN) با استفاده از واحد LSTM مطابق شکل زیر استفاده شده است. (مدل SimpleRNN در انتهای پاسخ سوال موجود است)



اطلاعات آماری این مدل نیز به صورت زیر می‌باشد:

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 20, 1)]           0

 lstm (LSTM)                 (None, 20, 50)            10400

 dropout (Dropout)           (None, 20, 50)            0

 lstm_1 (LSTM)               (None, 20, 50)            20200

 dropout_1 (Dropout)         (None, 20, 50)            0

 lstm_2 (LSTM)               (None, 20, 50)            20200

 dropout_2 (Dropout)         (None, 20, 50)            0

 lstm_3 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

مدل با استفاده از هایپر پارامترهای زیر آموزش داده شده است:

Loss function: Mean Squared Error (MSE)

Optimizer: Adam

Learning rate: 0.001

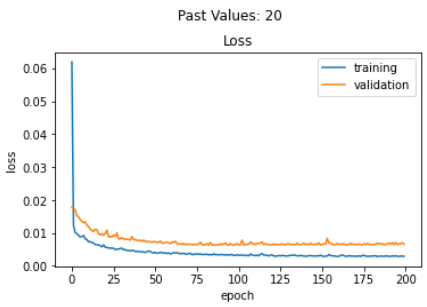
Epochs: 200

Batch size: 32

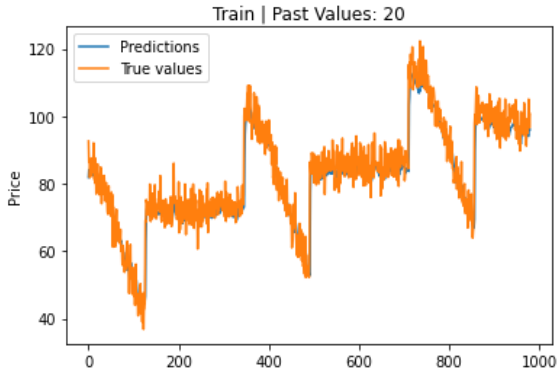
Past values window size: 20

```
2s 50ms/step - loss: 0.0029 - val_loss: 0.0066
```

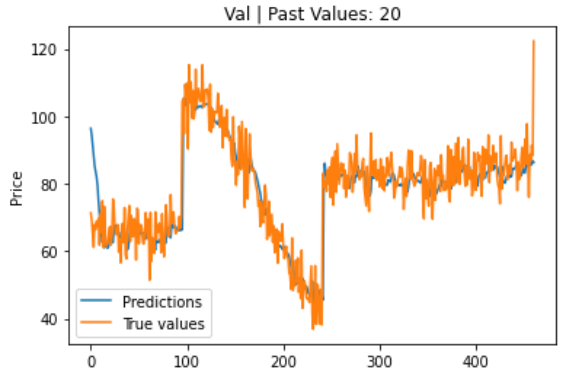
نمودار Loss در فاز آموزش و ارزیابی به صورت زیر می‌باشد:



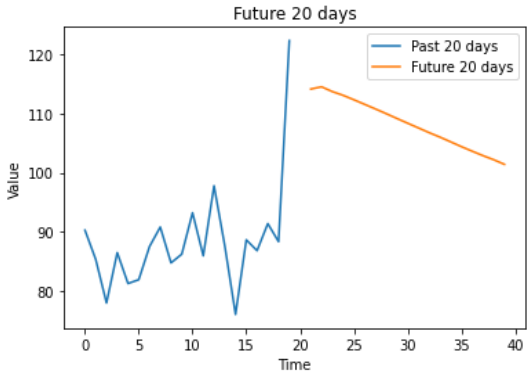
نمودار مقادیر واقعی فاز آموزش و پیش‌بینی مدل در این فاز در نمودار زیر موجود است:



همین نمودار بالا در فاز ارزیابی در شکل زیر آورده شده است:



همچنین به عنوان قسمت اختیاری که در صورت سوال ذکر نشده بود، مقادیر 20 روز آینده (پس از پایان دیتاست) نیز مطابق شکل زیر پیش‌بینی شده است:



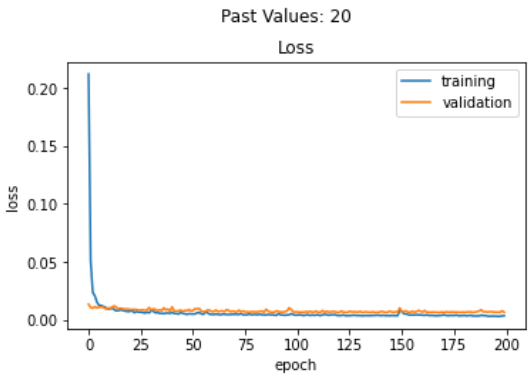
تمامی کارهای بالا بر روی یک مدل SimpleRNN نیز انجام گرفته که نتیجه آن در قالب تصاویر بیان می‌شود:

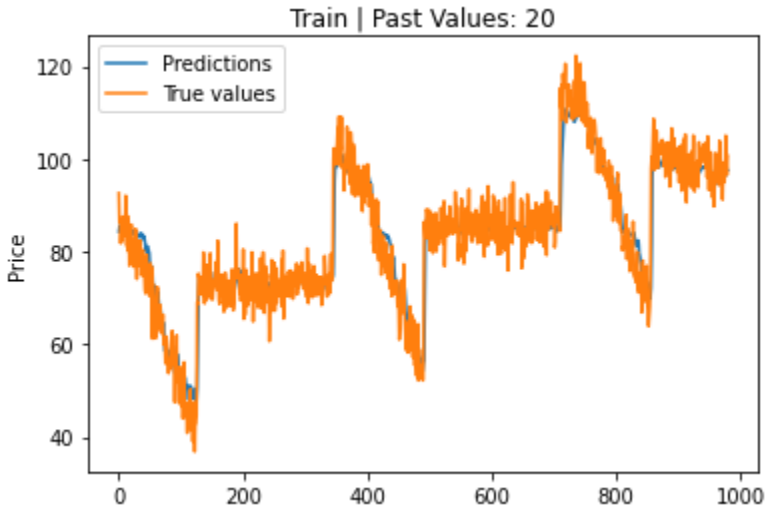
تعداد پارامترهای مدل از 71051 به 17801 کاهش یافته است. (تقریبا یک چهارم شده است.)

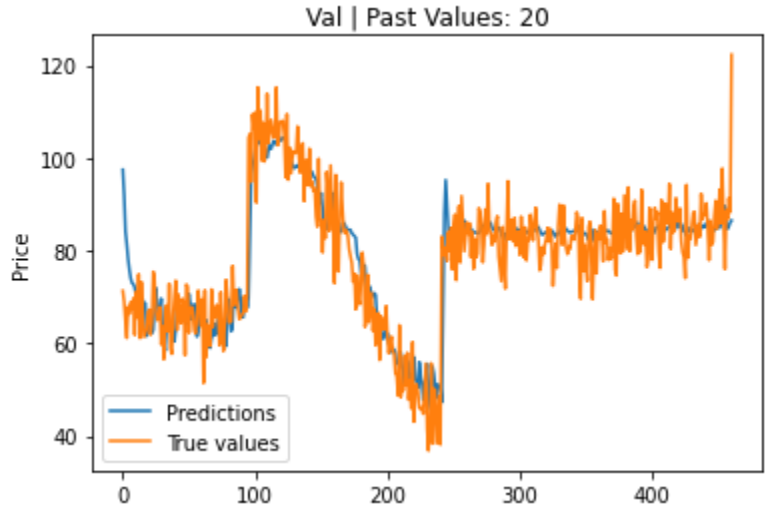
```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 20, 1)]           0

 simple_rnn (SimpleRNN)      (None, 20, 50)            2600

 dropout_6 (Dropout)         (None, 20, 50)            0

 simple_rnn_1 (SimpleRNN)    (None, 20, 50)            5050

 dropout_7 (Dropout)         (None, 20, 50)            0

 simple_rnn_2 (SimpleRNN)    (None, 20, 50)            5050

 dropout_8 (Dropout)         (None, 20, 50)            0

 simple_rnn_3 (SimpleRNN)    (None, 50)                5050

 dense_2 (Dense)             (None, 1)                 51

=================================================================
Total params: 17,801
Trainable params: 17,801
Non-trainable params: 0
_________________________________________________________________
```

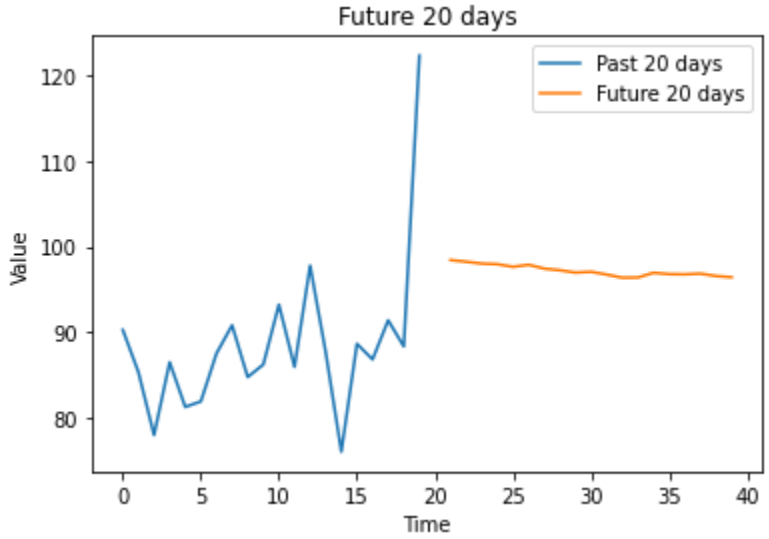
```
1s 24ms/step - loss: 0.0037 - val_loss: 0.0067
```

زمان آموزش در هر ایپاک (و هر بسته) تقریبا نصف شده است. خطا در حالت LSTM نسبت به SimpleRNN کمتر است.




Train | Past Values: 20
Predictions
True values
Price
120
100
80
60
40
0
200
400
600
800
1000


Val | Past Values: 20
Predictions
True values
Price
120
100
80
60
40
0
100
200
300
400


Future 20 days
Past 20 days
Future 20 days
Value
120
110
100
90
80
0
5
10
15
20
25
30
35
40
Time

Q2- GRU and Attention

Part A) GRU

ابتدا تابع Constructor کلاس MyGRUCell را مطابق زیر پیاده‌سازی می‌کنیم:

```
# ------------
# FILL THIS IN
# ------------
## Input linear layers
self.Wiz = nn.Linear(input_size, hidden_size, bias=False)
self.Wir = nn.Linear(input_size, hidden_size, bias=False)
self.Wih = nn.Linear(input_size, hidden_size, bias=False)

## Hidden linear layers
self.Whz = nn.Linear(hidden_size, hidden_size, bias=True)
self.Whr = nn.Linear(hidden_size, hidden_size, bias=True)
self.Whh = nn.Linear(hidden_size, hidden_size, bias=True)
```

سه تنسور اول مربوط به وزن‌های ورودی هستند و دارای Bias نیستند.

سه تنسور دومی مربوط به وزن‌های لایه مخفی (State) هستند و دارای Bias هستند.

به صورت کلی گیت‌های GRU به صورت زیر هستند:

Z: Update gate vector

R: Reset gate vector

H: Output vector

تابع Forward این کلاس نیز طبق معادلات درون صورت تمرین به صورت زیر پیاده‌سازی شده است:

$$r_t = \sigma(W_{ir}x_t + W_{hr}h_{t-1} + b_r) \quad (1)$$
$$z_t = \sigma(W_{iz}x_t + W_{hz}h_{t-1} + b_z) \quad (2)$$
$$g_t = \tanh(W_{in}x_t + r_t \odot (W_{hn}h_{t-1} + b_g)) \quad (3)$$
$$h_t = (1 - z) \odot g_t + z \odot h_{t-1}, \quad (4)$$

```
# ------------
# FILL THIS IN
# ------------
z = torch.sigmoid(self.Wiz(x) + self.Whz(h_prev))
r = torch.sigmoid(self.Wir(x) + self.Whr(h_prev))
g = torch.tanh(self.Wih(x) + r * self.Whh(h_prev))
h_new = (1 - z) * g + z * h_prev
return h_new
```

عملیات Backward نیز توسط ابزار مشتق گیری اتوماتیک PyTorch w صورت می‌گیرد و نیازی به پیاده‌سازی ما نیست.

Part B) Attention

گونه‌های مختلفی از مکانیزم Attention وجود دارد که در این تمرین می‌خواهیم گونه Additive آن را پیاده‌سازی کنیم.

ابتدا تابع Forward کلاس AdditiveAttention را مطابق شکل زیر پیاده‌سازی می‌کنیم:

```
# ------------
# FILL THIS IN
# ------------
batch_size, seq_len, hidden_size = keys.size()
expanded_queries = queries.unsqueeze(1).expand_as(keys)
concat_inputs = torch.cat((expanded_queries, keys), 2)
unnormalized_attention = self.attention_network(concat_inputs.view(-1, 2 * hidden_size)).view(batch_size, seq_len, 1)
attention_weights = self.softmax(unnormalized_attention)
context = torch.bmm(attention_weights.transpose(1, 2), values)
return context, attention_weights
```

این قسمت مطابق توضیحات فارسی و انگلیسی درون صورت تمرین پیاده‌سازی شده است.

تابع bmm دو ماتریس را به صورت batch شده در هم ضرب می‌کند.

سپس تابع Forward درون کلاس RNNAttentionDecoder را مطابق شکل زیر پیاده‌سازی می‌کنیم:

```
# ------------
# FILL THIS IN
# ------------
embed_current = embed[:, i, :]
context, attention_weights = self.attention(embed_current, annotations, annotations)
embed_and_context = torch.cat((embed_current, context.squeeze(1)), 1)
h_prev = self.rnn(embed_and_context, h_prev)
```

ابتدا بردار Context و وزن‌های Attention را با استفاده از تابع attention به دست آورده ایم.

سپس دو بردار Context و ورودی Decoder با هم تجمیع (Concat) شده است.

سپس با ورودی دادن بردار Concat شده و دادن آن به همراه H قبلی به شبکه RNN وضعیت جدید به دست می‌آید.

Results without attention

مدل را با هایپرپارامترهای زیر آموزش می‌دهیم:

```
            cuda: 1
         nepochs: 100
  checkpoint_dir: checkpoints
   learning_rate: 0.005
        lr_decay: 0.99
      batch_size: 64
     hidden_size: 20
    decoder_type: rnn
  attention_type:
```

نتایج ایپاک‌های آخر به صورت زیر است:

```
Epoch:  85 | Train loss: 0.800 | Val loss: 1.268 | Gen: ethay airway ongisioncay isspray orkinglay-estay
Epoch:  86 | Train loss: 0.789 | Val loss: 1.276 | Gen: ethay airway ongisioncay issway orklyway
Epoch:  87 | Train loss: 0.790 | Val loss: 1.182 | Gen: ethay airway ongingractingway isspay orkingday
Epoch:  88 | Train loss: 0.774 | Val loss: 1.199 | Gen: ethay airway ongisionicay-away ispay orkingday
Epoch:  89 | Train loss: 0.780 | Val loss: 1.224 | Gen: ethay airway ongiighingsray issray orkinggay
Epoch:  90 | Train loss: 0.764 | Val loss: 1.221 | Gen: ethay airway ongiighingsray issray orkingday
Epoch:  91 | Train loss: 0.760 | Val loss: 1.226 | Gen: ethay airway ongishingcay-ifedway isspay orkingway
Epoch:  92 | Train loss: 0.768 | Val loss: 1.256 | Gen: ethay airway ongiorinasecay isssay orkingday
Epoch:  93 | Train loss: 0.762 | Val loss: 1.164 | Gen: ethay airway ongingingfay issway orkinglay
Epoch:  94 | Train loss: 0.757 | Val loss: 1.242 | Gen: ethay airway ongiigingray issray orkingday
Epoch:  95 | Train loss: 0.762 | Val loss: 1.310 | Gen: ethay airway ongioningruredway isspray orkinglay-etatelycay
Epoch:  96 | Train loss: 0.769 | Val loss: 1.299 | Gen: ethay airway ongiisionisway issray orkingday
Epoch:  97 | Train loss: 0.768 | Val loss: 1.183 | Gen: ethay airway ongiighanceray isspay orkinglay
Epoch:  98 | Train loss: 0.758 | Val loss: 1.173 | Gen: ethay airway ongiighanisedway issway orkingday
Epoch:  99 | Train loss: 0.742 | Val loss: 1.166 | Gen: ethay airway ongisioncay isspay orkinglay
```

جمله اصلی و ترجمه شده نیز به صورت زیر است:

```
Source: the air conditioning is working
Translated: ethay airway ongisioncay isspay orkinglay
```

Results with attention

مدل را با هایپرپارامترهای زیر آموزش می‌دهیم:

```
            cuda: 1
         nepochs: 100
  checkpoint_dir: checkpoints
   learning_rate: 0.005
        lr_decay: 0.99
      batch_size: 64
     hidden_size: 20
    decoder_type: rnn_attention
  attention_type: additive
```

نتایج ایپاک‌های آخر به صورت زیر است:

```
Epoch:  72 | Train loss: 0.165 | Val loss: 0.612 | Gen: ehay airway onditionditiondionda isway orkingway-ingway-ing
Epoch:  73 | Train loss: 0.157 | Val loss: 0.570 | Gen: ethay airway onditinitingcay isway orkingway-orkingway
Epoch:  74 | Train loss: 0.145 | Val loss: 0.416 | Gen: ethay airway onditingcay isway orkingway
Epoch:  75 | Train loss: 0.100 | Val loss: 0.381 | Gen: ethay airway onditingcay isway orkingway
Epoch:  76 | Train loss: 0.092 | Val loss: 0.400 | Gen: ethay airway onditingcay isway orkingway
Epoch:  77 | Train loss: 0.084 | Val loss: 0.374 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  78 | Train loss: 0.080 | Val loss: 0.376 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  79 | Train loss: 0.077 | Val loss: 0.373 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  80 | Train loss: 0.075 | Val loss: 0.373 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  81 | Train loss: 0.074 | Val loss: 0.376 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  82 | Train loss: 0.073 | Val loss: 0.377 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  83 | Train loss: 0.072 | Val loss: 0.377 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  84 | Train loss: 0.071 | Val loss: 0.380 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  85 | Train loss: 0.070 | Val loss: 0.388 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  86 | Train loss: 0.070 | Val loss: 0.392 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  87 | Train loss: 0.069 | Val loss: 0.402 | Gen: ethay airway onditingcay isway orkingway
Epoch:  88 | Train loss: 0.069 | Val loss: 0.410 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  89 | Train loss: 0.068 | Val loss: 0.400 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  90 | Train loss: 0.069 | Val loss: 0.502 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  91 | Train loss: 0.088 | Val loss: 0.437 | Gen: ethay airway onditionditiondicay isway orkingway
Epoch:  92 | Train loss: 0.079 | Val loss: 0.413 | Gen: ethay airway onditingcay isway orkingway
Epoch:  93 | Train loss: 0.072 | Val loss: 0.394 | Gen: ethay airway onditingcay isway orkingway
Epoch:  94 | Train loss: 0.066 | Val loss: 0.417 | Gen: ethay airway onditingcay isway orkingway
Epoch:  95 | Train loss: 0.064 | Val loss: 0.412 | Gen: ethay airway onditinitionditionda isway orkingway
Epoch:  96 | Train loss: 0.065 | Val loss: 0.438 | Gen: ethay airway onditinitingcay isway orkingway
Epoch:  97 | Train loss: 0.068 | Val loss: 0.475 | Gen: ethay airway onvcay isway orkingway
Epoch:  98 | Train loss: 0.082 | Val loss: 0.533 | Gen: esthay airway onditingcay isway orkingway
Epoch:  99 | Train loss: 0.108 | Val loss: 0.568 | Gen: ethay airway onditinitingcay isway orkway
```

همانطور که مشاهده می‌شود در حالت Attention میزان خطا هم در آموزش و هم ارزیابی بسیار بسیار کمتر بوده.

جمله اصلی و ترجمه شده نیز به صورت زیر است:

```
Source: the air conditioning is working
Translated: ethay airway onditinitingcay isway orkway
```

همانطور که مشاهده شد کلمات ethay و airway در دو حالت به درستی خروجی داده شده اند. اما کلمه conditioning در حالت بدون مکانیزم توجه به اشتباه ongisioncay نوشته شده است در حالی که املای درست آن onditinitingcay می‌باشد که در حالت با مکانیزم توجه به درستی خروجی داده شده است.

کلمه is نیز در حالت اول به اشتباه isspay خروجی داده شده است اما در حالت با توجه به درستی isway ترجمه شده است. کلمه working نیز در حالت اول به اشتباه orkinglay ترجمه شده اما در حالت اتنشن به درستی orkway ترجمه شده است.

در واقع مکانیزم توجه این امکان را فراهم می‌کند که Decoder توجه خود را به بخش‌های مهمتر Encoder یا (حتی خودش) معطوف کند و اطلاعات فراموش نشوند. در این تسک که در سطح کلمات پیش‌بینی صورت می‌گرفت اثر آن به خوبی قابل دیدن بود.