بسمه تعالی

دانشکده مهندسی کامپیوتر

یادگیری عمیق

نام استاد: دکتر محمدی

تمرین دوازدهم

آرمان حیدری

شماره دانشجویی: ۹۷۵۲۱۲۵۲

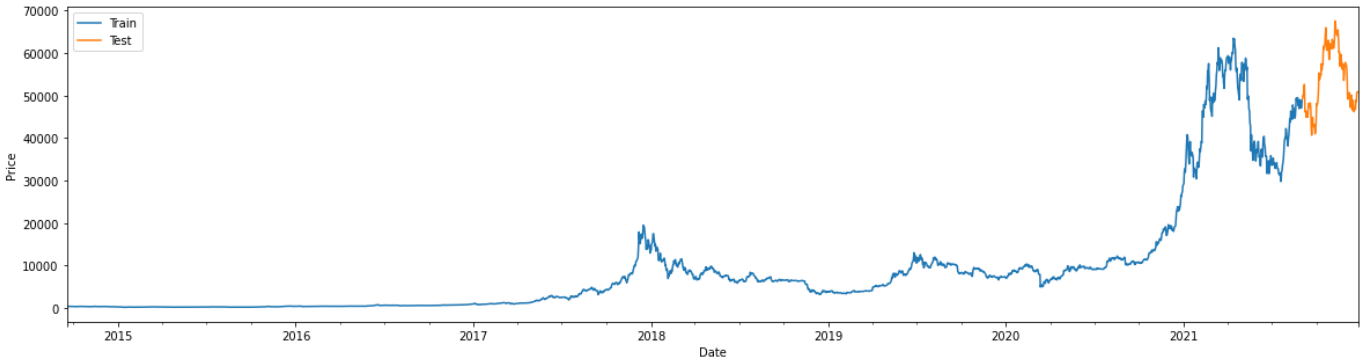
آذر ۱۴۰۰

پاسخ سوال اول

مراحل مختلف این سوال در نوتبوک HW12.ipynb (لینک گوگل کولب)، قسمت question 1 پیاده سازی شده است.

توضیحات مرحله به مرحله ی کد، بین سلول های نوتبوک داده شده اند. که شامل لود کردن دیتاست و نمایش قسمت آموزشی و آزمایشی، پیش پردازش داده، تعریف مدل و آموزش آن و در نهایت دیدن نتیجه روی داده های آزمیشی است.

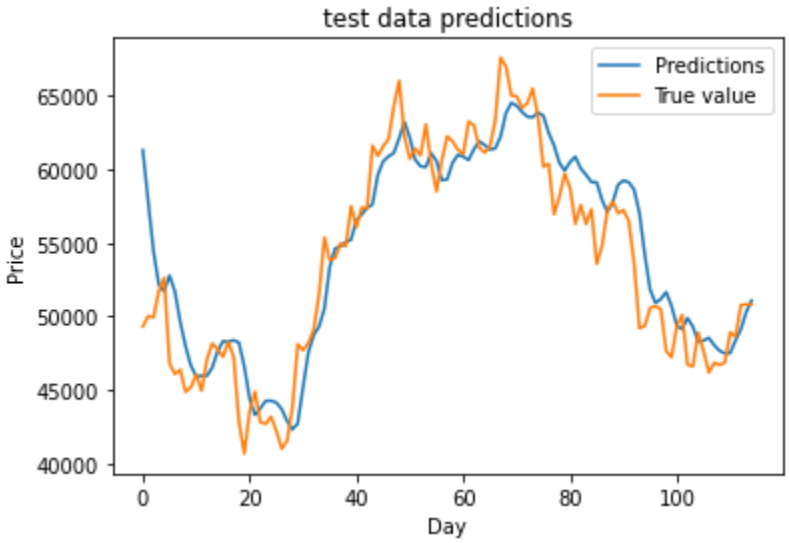
نمودار داده های آموزشی و آزمایشی به این صورت است:



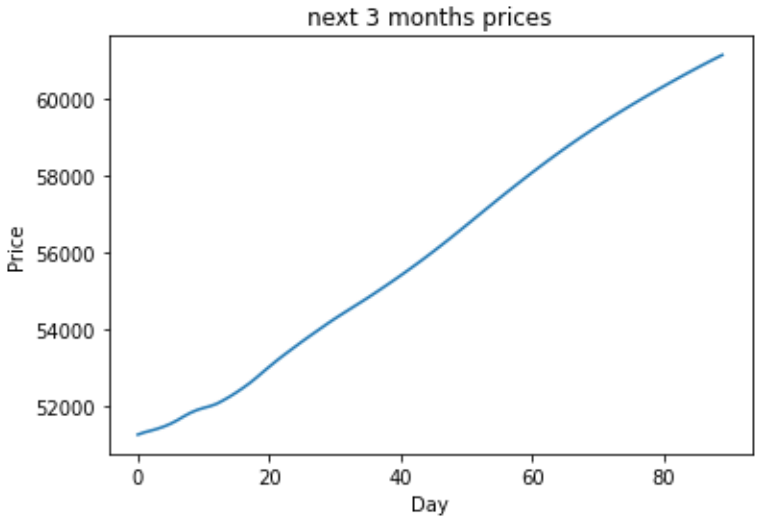
در اینجا چون معیار دقیقی برای accuracy تعریف نمیکنیم و طبیعتا نمیتواند قیمت را با دقت ۴ رقم اعشار پیشبینی کند و به همین دلیل نتایج accuracy غیر منطقی خواهند بود، با همان سنجیدن loss و روند کاهشی مطلوبی که دارد میفهمیم که مدل به خوبی آموزش دیده است:

```
Epoch 1/100
80/80 [==============================] - 22s 61ms/step - loss: 0.0050
Epoch 2/100
80/80 [==============================] - 5s 60ms/step - loss: 0.0018
Epoch 3/100
80/80 [==============================] - 5s 60ms/step - loss: 0.0013
Epoch 4/100
80/80 [==============================] - 5s 61ms/step - loss: 0.0016
Epoch 5/100
80/80 [==============================] - 5s 61ms/step - loss: 0.0013
Epoch 6/100
80/80 [==============================] - 5s 61ms/step - loss: 0.0010
Epoch 7/100
80/80 [==============================] - 5s 60ms/step - loss: 9.9566e-04
Epoch 8/100
80/80 [==============================] - 5s 60ms/step - loss: 8.5756e-04
Epoch 9/100
80/80 [==============================] - 5s 60ms/step - loss: 7.2087e-04
Epoch 10/100
80/80 [==============================] - 5s 61ms/step - loss: 6.8993e-04
Epoch 11/100
80/80 [==============================] - 5s 61ms/step - loss: 9.9726e-04
Epoch 12/100
80/80 [==============================] - 5s 61ms/step - loss: 6.6609e-04
Epoch 13/100
80/80 [==============================] - 5s 64ms/step - loss: 6.1974e-04
Epoch 14/100
80/80 [==============================] - 5s 61ms/step - loss: 6.5841e-04
```

و نتیجه پیشبینی مدل برای ۱۱۵ روز داده تست هم در مقایسه با داده های واقعی به این صورت بود:



که میبینیم نتایج معقولی است و شکل کلی حرکت قیمت را تا حد خوبی پیشبینی کرده است. برای سه ماه آینده هم با استفاده از مدل پیشبینی میکنیم که به این صورت به دست آمد:



میبینیم که نمودار غیرواقعی به نظر میرسد. به دلیل این است که این شبکه برای داده unsupervised چندان مناسب نیست و نتیجه ای که در هر روز بر اساس ۶۰ روز گذشته گرفته فقط بر اساس قیمت هایی است که داریم و همواره عددی کمی بیشتر از قبل به دست می آید و نوسانات را پیشبینی نمیکند.

به نظر شما افزایش یا کاهش متغیر تعریف شده در مرحله تهیه داده مورد نیاز برای آموزش مدل یعنی تعداد داده های گذشته برای پیشبینی داده های مشخص چه مزایا یا معایبی دارد؟ شرح دهید.

این متغیر که در کد من به اسم PAST_VALUES تعریف شده است، پارامتر مهمی در این مسئله است. چون هرچقدر مقدار آن را کمتر کنیم، درواقع ویژگی هایی کمتری را برای پیشبینی در اختیار مدل قرار داده ایم و مزیتش قطعا سرعت بیشتر مدل خواهد بود. چون ابعاد ورودی به لایه LSTM کاش میابد و به خصوص چون در شبکه های بازگشتی محاسبات ناچارا ترتیبی هستند و نه موازی، این تغییر محسوس خواهد بود. عیب اصلی کم کردن این عدد هم قطعا کم شدن دقت (افزایش ضرر) خواهد بود. چون به هرحال به مدل داده کمتری را داده ایم.

با زیاد کردن این پارامتر، سرعت شبکه کاهش میابد و دقت هم بیشتر می شود. البته باید دقت داشته باشیم که خیلی زیاد کردن آن حتی میتواند در برخی مسائل مدل را به اشتباه هم بیندازد. چون مثلا قیمت ۱ سال پیش بیت کوین احتمالا اهمیت چندانی برای پیشبینی قیمت فردا نخواهد داشت. و همچنین زیادی بزرگ شدن شبکه میتواند باعن پدیده gradient vanishing شود، که البته به علت استفاده ما از LSTM به جای simple RNN این قضیه کمتر خواهد بود.

منابع

https://seaborn.pydata.org/generated/seaborn.lineplot.html

https://compucademy.net/getting-stock-data-using-python-and-yfinance/

پاسخ سوال دوم

مراحل مختلف این سوال در نوتبوک HW12.ipynb (لینک گوگل کولب)، قسمت question 2 پیاده سازی شده است.

در ابتدا برای ذخیره کردن مدل، google drive را mount میکنیم تا وزن ها و سایر موارد مدل ها را پس از هر epoch، در یک فایل my_model.h5 داخل دایرکتوری DL_HW12 ذخیره کنیم. این ذخیره کردن را با استفاده از تابع save(). که در کراس برای مدل ها پیاده سازی شده است انجام داده ایم. برای خواندن مدل ذخیره شده هم، در هنگام تست کردن و بعضا اگر آموزش نصفه ماند و خواستیم ادامه دهیم، میتوانیم از تابع load_model پیاده سازی شده در keras.models استفاده کنیم.

در این مسئله با مدل sequence to sequence سروکار داریم. و درواقع با یک لایه GRU به عنوان many to one و یک یا چند لایه GRU به عنوان one to many استفاده میکنیم. تعداد واحدهای هر لایه میتواند هر عددی باشد اما طبق مثال کلاسی حدودا ۶۴ و ۱۲۸ و ۲۵۶ را امتحان میکنیم تا ببینیم کدام نتایج بهتری دارد.

با پیش‌پردازش روی داده های ورودی آن ها را به صورت one hot coding در می‌آوریم. داده تست را هم ۱۰ تا ۱۰ تا جدا کرده و به همین صورت در میآوریم تا بتوانیم به مدل تعریف شده بدهیم و خروجی بگیریم. توابع مختلف در نوتبوک توضیح داده شده اند. هایپرپارامترهای مسئله را به این صورت تعریف میکنیم:

```
optimizer = 'adam'
loss = 'categorical_crossentropy'
epochs = 10
batch_size=32
save_path = '/content/gdrive/MyDrive/DL_HW12/my_model.h5'
```

مدل اول به همراه نتیجه روی داده آموزشی و جمله تست:

```
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 128)               60288

repeat_vector (RepeatVector  (None, 10, 128)           0
)

gru_1 (GRU)                  (None, 10, 128)           99072

dense (Dense)                (None, 10, 27)            3483
```

```
4759/4759 [==============================] - 61s 11ms/step - loss: 2.2315 - accuracy: 0.3362
4759/4759 [==============================] - 54s 11ms/step - loss: 1.3802 - accuracy: 0.5714
4759/4759 [==============================] - 52s 11ms/step - loss: 0.6713 - accuracy: 0.7894
4759/4759 [==============================] - 52s 11ms/step - loss: 0.3521 - accuracy: 0.8956
4759/4759 [==============================] - 53s 11ms/step - loss: 0.2412 - accuracy: 0.9292
4759/4759 [==============================] - 53s 11ms/step - loss: 0.1954 - accuracy: 0.9425
4759/4759 [==============================] - 54s 11ms/step - loss: 0.1728 - accuracy: 0.9488
4759/4759 [==============================] - 54s 11ms/step - loss: 0.1584 - accuracy: 0.9527
4759/4759 [==============================] - 53s 11ms/step - loss: 0.1491 - accuracy: 0.9553
4759/4759 [==============================] - 53s 11ms/step - loss: 0.1416 - accuracy: 0.9573

test_model(load_model(save_path))

    k       love     deep     slearring
```

در مدل دوم تعداد unit هارا از ۱۲۸ به ۲۵۶ افزایش دادیم. میبینیم که آموزش کمی بهتر انجام شده است که به علت پارامترهای بیشتر مدل طبیعیست. نتیجه روی جمله تست اما تغییر کیفیت چندانی نداشته است:

```
Layer (type)                 Output Shape              Param #
=================================================================
gru_2 (GRU)                  (None, 256)               218880

repeat_vector_1 (RepeatVect  (None, 10, 256)           0
or)

gru_3 (GRU)                  (None, 10, 256)           394752

dense_1 (Dense)              (None, 10, 27)            6939
```

```
4759/4759 [==============================] - 68s 13ms/step - loss: 1.7370 - accuracy: 0.4802
4759/4759 [==============================] - 64s 13ms/step - loss: 0.3633 - accuracy: 0.8917
4759/4759 [==============================] - 65s 14ms/step - loss: 0.1978 - accuracy: 0.9407
4759/4759 [==============================] - 64s 13ms/step - loss: 0.1609 - accuracy: 0.9511
4759/4759 [==============================] - 63s 13ms/step - loss: 0.1437 - accuracy: 0.9558
4759/4759 [==============================] - 64s 13ms/step - loss: 0.1329 - accuracy: 0.9591
4759/4759 [==============================] - 63s 13ms/step - loss: 0.1261 - accuracy: 0.9609
4759/4759 [==============================] - 63s 13ms/step - loss: 0.1199 - accuracy: 0.9628
4759/4759 [==============================] - 64s 13ms/step - loss: 0.1156 - accuracy: 0.9639
4759/4759 [==============================] - 63s 13ms/step - loss: 0.1112 - accuracy: 0.9653

test_model(load_model(save_path))

   bi       love    redeep     alearning
```

حالا سعی میکنیم با عمیقتر کردن شبکه به نتایج بهتری برسیم. البته در شبکه های RNN مانند CNN صحبت از صد لایه نمیکنیم و معمولا زیر ۵ لایه هستند. لایه dense آخر را هرگز تغییر نمیدهیم چون برای تصمیم گیری

نهایی بین ۲۷ کاراکتر (۲۶ حرف انگلیسی+space) است و به همین دلیل تابع فعالسازی softmax دارد. نتایج مدل سوم:

Layer (type)	Output Shape	Param #
gru_7 (GRU)	(None, 128)	60288
repeat_vector_3 (RepeatVector)	(None, 10, 128)	0
gru_8 (GRU)	(None, 10, 128)	99072
gru_9 (GRU)	(None, 10, 128)	99072
dense_3 (Dense)	(None, 10, 27)	3483

```
4759/4759 [==============================] - 81s 15ms/step - loss: 2.1821 - accuracy: 0.3453
4759/4759 [==============================] - 72s 15ms/step - loss: 1.2724 - accuracy: 0.5855
4759/4759 [==============================] - 72s 15ms/step - loss: 0.6474 - accuracy: 0.7870
4759/4759 [==============================] - 72s 15ms/step - loss: 0.4586 - accuracy: 0.8508
4759/4759 [==============================] - 71s 15ms/step - loss: 0.3705 - accuracy: 0.8803
4759/4759 [==============================] - 72s 15ms/step - loss: 0.3166 - accuracy: 0.8984
4759/4759 [==============================] - 71s 15ms/step - loss: 0.2815 - accuracy: 0.9104
4759/4759 [==============================] - 79s 17ms/step - loss: 0.2531 - accuracy: 0.9203
4759/4759 [==============================] - 72s 15ms/step - loss: 0.2323 - accuracy: 0.9272
4759/4759 [==============================] - 71s 15ms/step - loss: 0.2169 - accuracy: 0.9324
```

```
test_model(load_model(save_path))
```

```
    i       love     deep     alearning
```

علی رغم این که مدل دیرتر همگرا شده است و حتی از مدل اول هم دقتش روی داده آموزشی کمتر است، به نتیجه ای عالی روی جمله تست میرسد. و استفاده از دولایه با ۱۲۸ واحد در قسمت one to many به نظر به نتیجه خوبی میرسد.

در مدل چهارم سعی کردم با ۶۴ واحد در ۳ لایه GRU مسئله را حل کنم اما نتیجه روی داده آموزشی و تست جالب نبود:

Layer (type)	Output Shape	Param #
gru_10 (GRU)	(None, 64)	17856
repeat_vector_4 (RepeatVector)	(None, 10, 64)	0
gru_11 (GRU)	(None, 10, 64)	24960
gru_12 (GRU)	(None, 10, 64)	24960
dense_4 (Dense)	(None, 10, 27)	1755

```
4759/4759 [==============================] - 69s 14ms/step - loss: 2.3352 - accuracy: 0.3042
4759/4759 [==============================] - 65s 14ms/step - loss: 2.0296 - accuracy: 0.3764
4759/4759 [==============================] - 64s 14ms/step - loss: 1.7973 - accuracy: 0.4455
4759/4759 [==============================] - 64s 14ms/step - loss: 1.5599 - accuracy: 0.5093
4759/4759 [==============================] - 64s 14ms/step - loss: 1.1920 - accuracy: 0.6086
4759/4759 [==============================] - 64s 13ms/step - loss: 0.8795 - accuracy: 0.7038
4759/4759 [==============================] - 64s 13ms/step - loss: 0.7179 - accuracy: 0.7546
4759/4759 [==============================] - 64s 14ms/step - loss: 0.6134 - accuracy: 0.7910
4759/4759 [==============================] - 65s 14ms/step - loss: 0.5389 - accuracy: 0.8186
4759/4759 [==============================] - 65s 14ms/step - loss: 0.4835 - accuracy: 0.8386
```

```
test_model(load_model(save_path))
```

```
s  wa        lave    swdeer    plearning
```