گزارش پروژه درس

یادگیری ماشین

استاد درس

دکتر محمدزاده

دانشگاه صنعتی شریف

دانشکده مهندسی برق

آرمان بختیاری

93103809

مقدمه

هدف به دست آوردن مدلی است که بتواند قیمت ارز/کالا را برای 5، 10، ... و50 دقیقه آینده پیش بینی کند. برای این منظور استفاده از اطلاعات مربوط به فایل های A_ticker, B_ticker, C_ticker کافی است. همچنین برای طراحی این مدل از LSTM از پکیج keras استفاده شده است که بعد تر در مورد آن توضیحاتی ارایه خواهم داد.

مرحله پیش پردازش (pre-processing) :

پس از خواندن یکی از فایل های موسوم به ticker، ابتدا با توجه اینکه اگر داده ای غلط گزارش شده باشد کل سطر مربوطه 1- میشود، باید این سطور حذف شوند تا تاثیر چندانی در طراحی مدل نداشته باشد. این عمل را با جایگزینی سطری که 1- است با سطر پیشین آن به شکل روبه رو انجام میدهیم.

```
for file_name in ('A_ticker.csv', 'B_ticker.csv', 'C_ticker.csv'):
    temp = []
    with open(file_name) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        for row in csv_reader:
            for i in range(0, 9):
                row[i] = float(row[i])
                # Eliminating -1s
                if row[i] == -1:
                    row[i] = row_temp[i]
            row_temp = row
            temp.append(row)
```

حال ماتریس d را که سطر های آن اصلاح شده اند را تشکیل داده و آن را نورمالیزه میکنیم. حاصل را در ماتریسی به نام data ذخیره می کنیم.

```
d = np.array(temp)
# Normalization
scaler = MinMaxScaler(feature_range=(0, 1))
data = scaler.fit_transform(d)
```

حال لیبل داده ها را که همان آخرین قیمت معامله شده است (ستون ششم) مشخص کرده و داده های تست و ترین را از هم جدا میکنیم.
توجه باید کرد که مثلا قیمت دقیقه 10 همان مقداری است که باید برای داده دقیقه 5 پیش بینی شود، نتیجتا لیبل ها باید یک شیفت بخورند. به همین ترتیب تمام لیبل داده های تست را که شیفت خورده ی لیبل 5 دقیقه بعدی است تعیین میکنیم.

```
labels = data[:, 5]
for p in range(10499):
    labels[p] = labels[p+1]
# Splitting Train and Test data and labels
train_data = data[0:8000, 0:9]
train_labels = labels[0:8000]

test_data = data[8000:10500, 0:9]
test_labels = labels[8000:10500]
```

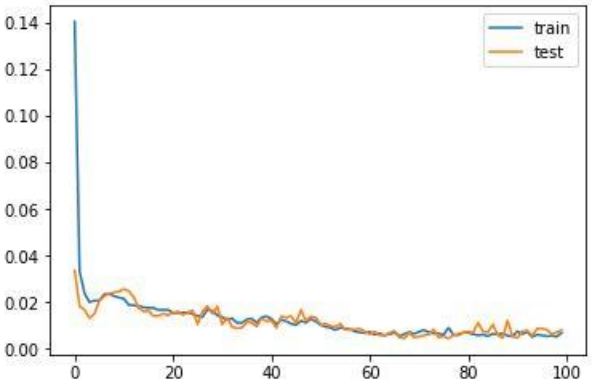
```
# Test labels for 10, 15,...,50 minutes after test time

temp_labels = test_labels

test_labels10 = np.zeros(2500)
for p in range(2499):
    test_labels10[p] = temp_labels[p+1]
test_labels10[2499] = test_labels10[2498]
test_labels15 = np.zeros(2500)
for p in range(2499):
    test_labels15[p] = test_labels10[p+1]
test_labels15[2499] = test_labels15[2498]
test_labels20 = np.zeros(2500)
for p in range(2499):
    test_labels20[p] = test_labels15[p+1]
test_labels20[2499] = test_labels20[2498]
test_labels25 = np.zeros(2500)
for p in range(2499):
    test_labels25[p] = test_labels20[p+1]
test_labels25[2499] = test_labels25[2498]
test_labels30 = np.zeros(2500)
for p in range(2499):
    test_labels30[p] = test_labels25[p+1]
test_labels30[2499] = test_labels30[2498]
test_labels35 = np.zeros(2500)
for p in range(2499):
    test_labels35[p] = test_labels30[p+1]
test_labels35[2499] = test_labels35[2498]
test_labels40 = np.zeros(2500)
for p in range(2499):
    test_labels40[p] = test_labels35[p + 1]
test_labels40[2499] = test_labels40[2498]
test_labels45 = np.zeros(2500)
for p in range(2499):
    test_labels45[p] = test_labels40[p+1]
test_labels45[2499] = test_labels45[2498]
test_labels50 = np.zeros(2500)
for p in range(2499):
    test_labels50[p] = test_labels45[p+1]
test_labels50[2499] = test_labels50[2498]
```

ساخت مدل :

همانطور که گفته شد از مدل LSTM در این پروژه استفاده شده که در پایان گزارش اجمالا به توضیح آن میپردازیم.

برای استفاده از مدل LSTM ابتدا باید داده های ترین را سه بعدی کنیم. سپس LSTM را با 50 لایه پنهان، یک لایه خروجی و آپتیمایزر adam لرن میکنیم. سپس مدل را بر روی داده های خود برای 100 دوره تکرار fit میکنیم

```
# Reshaping data to suitable format (3d) for learning
train_data = train_data.reshape((train_data.shape[0], 1, train_data.shape[1]))
test_data = test_data.reshape((test_data.shape[0], 1, test_data.shape[1]))
print(train_data.shape, train_labels.shape, test_data.shape, test_labels.shape)
# defining learning model
model = Sequential()
model.add(LSTM(50, input_shape=(train_data.shape[1], train_data.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
# fit network
history = model.fit(train_data, train_labels, epochs=100, batch_size=72, validation_data=(test_data, te
# plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```



پیش بینی (prediction) :

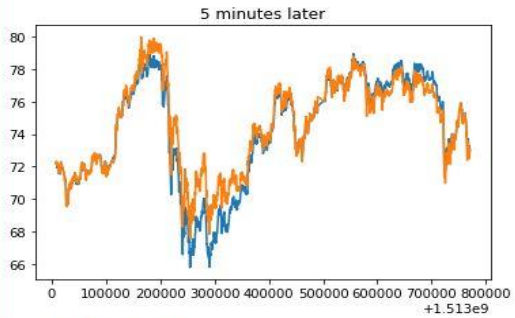
حال آنچه که هدف پروژه بود، یعنی پیش بینی قیمت کالا در دقایق خواسته شده، را بر اساس داده های تست مان بدست می آوریم. در اینجا چند نکته شایان ذکر است. اولا اساس کار مدل پیش بینی قیمت پنج دقیقه بعد بر مبنای داده های ورودی آن است. یعنی مثلا ما برای پیش بینی قیمت دقیقه 10، از پیش بینی مربوط به دقیقه 5 استفاده میکنیم و به همین ترتیب تا آخر. دوما چون لایه خروجی LSTM یعنی Dense دو بعدی است از تابع reshape برای اصلاح آن استفاده میکنیم. سوما برای دینورمایزه کردن داده ها خروجی را در عدد 80 ضرب کرده ایم که البته کار دقیقی نیست. در نهایت rmse را برای هر کدام از دقایق خواسته شده نسبت به داده های تست بدست می آوریم.

```
# Predicting test data
yhat5 = model.predict(test_data)
t = d[8000:10500, 0]

yhat5 = yhat5.reshape(2500)

# Evaluating and Comparing predicted labels and original lables for test data
pyplot.plot(t, yhat5*80)
pyplot.plot(t, test_labels*80)
pyplot.title('5 minutes later')

pyplot.show()

# Predicting prices for other times (10, 15, ...,50 mins from now)

test_data[0:2500, 0, 5] = yhat5

yhat10 = model.predict(test_data)
yhat10 = yhat10.reshape(2500)
test_data[0:2500, 0, 5] = yhat10
```

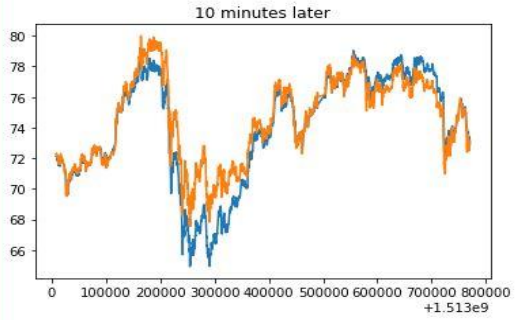
```
rmse10 = sqrt(mean_squared_error(yhat10, test_labels10))
print('Test RMSE: %.3f' % rmse10)
pyplot.plot(t, yhat10*80)
pyplot.plot(t, test_labels10*80)
pyplot.title('10 minutes later')

pyplot.show()

yhat15 = model.predict(test_data)
yhat15 = yhat15.reshape(2500)
test_data[0:2500, 0, 5] = yhat15

pyplot.plot(t, yhat15*80)
pyplot.plot(t, test_labels15*80)
pyplot.title('15 minutes later')

pyplot.show()

yhat20 = model.predict(test_data)
yhat20 = yhat20.reshape(2500)
test_data[0:2500, 0, 5] = yhat20

pyplot.plot(t, yhat20*80)
pyplot.plot(t, test_labels20*80)
pyplot.title('20 minutes later')

pyplot.show()

yhat25 = model.predict(test_data)
yhat25 = yhat25.reshape(2500)
test_data[0:2500, 0, 5] = yhat25

pyplot.plot(t, yhat25*80)
pyplot.plot(t, test_labels25*80)
pyplot.title('25 minutes later')

pyplot.show()
```

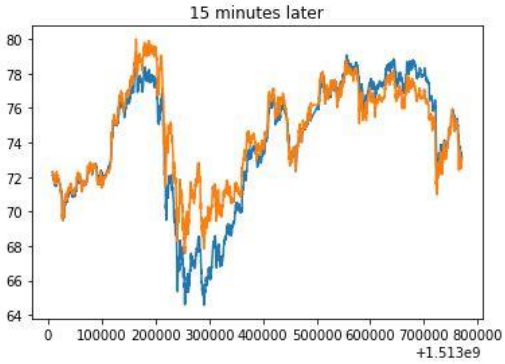
توجه : تمام کد را

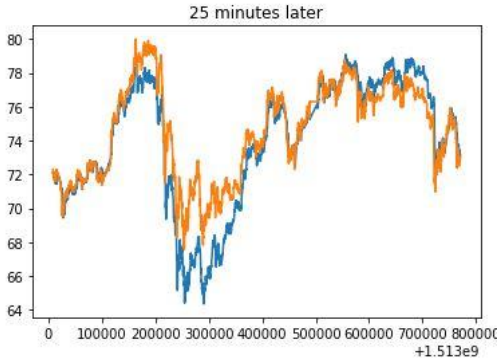
در اینجا نیاوردیم !

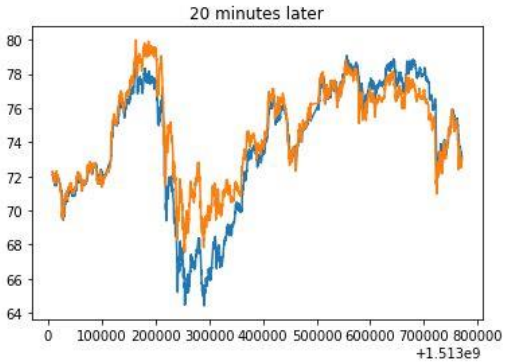


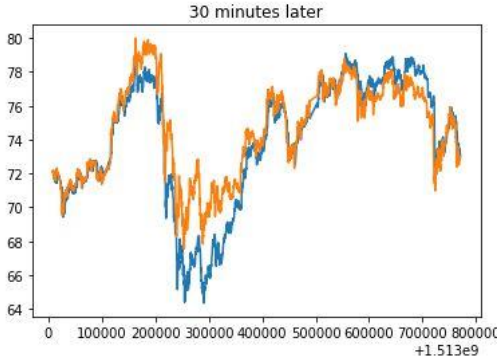

```
Test RMSE: 0.017
```

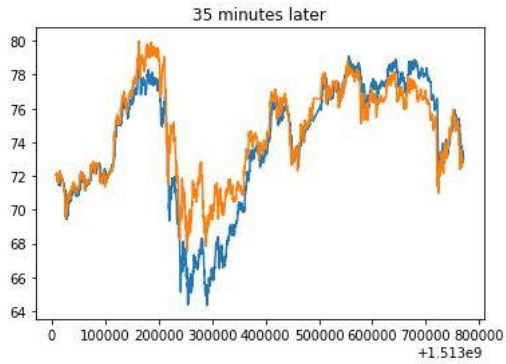


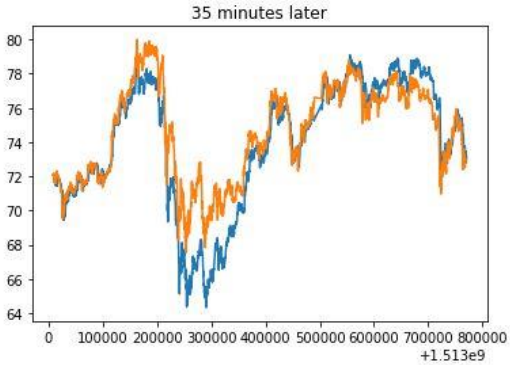

15 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9

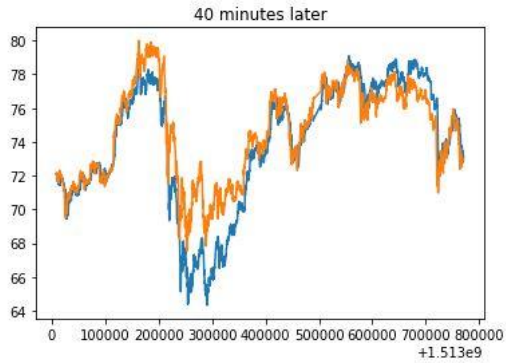

25 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9

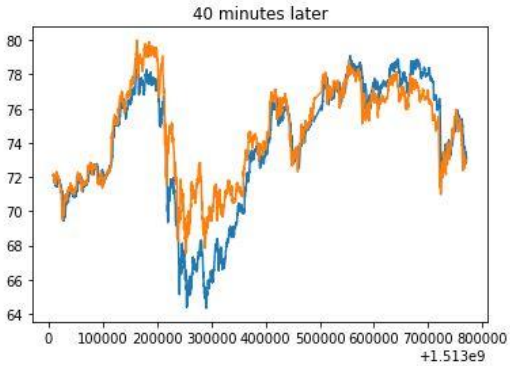

20 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9


30 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9


35 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9

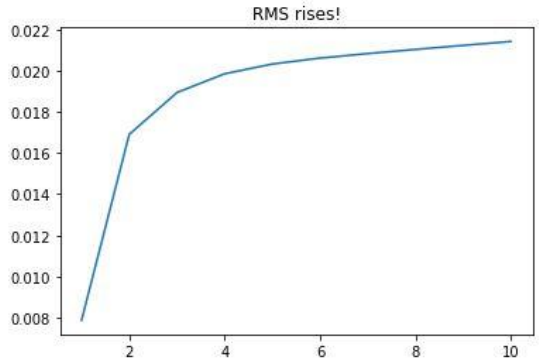

35 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9


40 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9


40 minutes later
80
78
76
74
72
70
68
66
64
0 100000 200000 300000 400000 500000 600000 700000 800000
+1.513e9

```
rmse5 = sqrt(mean_squared_error(yhat5, test_labels))
print('5 mins later==> Test RMSE: %.7f' % rmse5)

rmse10 = sqrt(mean_squared_error(yhat10, test_labels10))
print('10 mins later==> Test RMSE: %.7f' % rmse10)

rmse15 = sqrt(mean_squared_error(yhat15, test_labels15))
print('15 mins later==> Test RMSE: %.7f' % rmse15)

rmse20 = sqrt(mean_squared_error(yhat20, test_labels20))
print('20 mins later==> Test RMSE: %.7f' % rmse20)

rmse25 = sqrt(mean_squared_error(yhat25, test_labels25))
print('25 mins later==> Test RMSE: %.7f' % rmse25)

rmse30 = sqrt(mean_squared_error(yhat30, test_labels30))
print('30 mins later==> Test RMSE: %.7f' % rmse30)

rmse35 = sqrt(mean_squared_error(yhat35, test_labels35))
print('35 mins later==> Test RMSE: %.7f' % rmse35)

rmse40 = sqrt(mean_squared_error(yhat40, test_labels40))
print('40 mins later==> Test RMSE: %.7f' % rmse40)

rmse45 = sqrt(mean_squared_error(yhat45, test_labels45))
print('45 mins later==> Test RMSE: %.7f' % rmse45)

rmse50 = sqrt(mean_squared_error(yhat50, test_labels50))
print('50 mins later==> Test RMSE: %.7f' % rmse50)

RMSs = [rmse5, rmse10, rmse15, rmse20, rmse25, rmse30, rmse35, rmse40, rmse45, rmse50]
pyplot.plot([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], RMSs)
pyplot.title('RMS rises!')
pyplot.show()
```

```
5 mins later==> Test RMSE: 0.0078753
10 mins later==> Test RMSE: 0.0169004
15 mins later==> Test RMSE: 0.0189332
20 mins later==> Test RMSE: 0.0198462
25 mins later==> Test RMSE: 0.0203198
30 mins later==> Test RMSE: 0.0206073
35 mins later==> Test RMSE: 0.0208267
40 mins later==> Test RMSE: 0.0210284
45 mins later==> Test RMSE: 0.0212234
50 mins later==> Test RMSE: 0.0214144
```

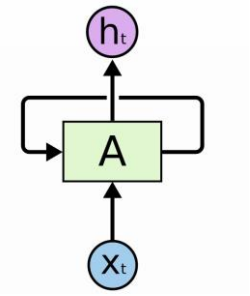


LSTM (Long-Short Term Memory)

در ابتدا باید به شبکه های عصبی بازگشتی (RNN) بپردازیم.

شبکه‌های عصبی بازگشتی(RNN)

در مورد نحوه فکر کردن انسان‌ها میشود گفت که اینطور نیست که هر ثانیه ریست شود. در همین لحظه که دارید این مطلب را می‌خوانید شما معنی هر کلمه را با توجه به دانشی که از خواندن کلمات قبلی کسب کرده اید متوجه میشوید. به عبارتی شما موقع خواندن یک متن، درک و فهمی را که در مورد آن متن با توجه به خواندن کلمات قبل کسب کردید دور نمی‌ریزید بلکه به صورت پیوسته با خواندن هر کلمه جدید، نسبت به آن متنی که میخوانید درک و فهم پیدا می کنید و به عبارتی معنی آن متن را متوجه میشوید. تا قبل از شبکه های RNN، شبکه های عصبی متداول چنین خاصیتی نداشتند. شبکه‌های عصبی بازگشتی برای برطرف کردن این مشکل طراحی شدند. در حقیقت شبکه‌های عصبی بازگشتی در خود شامل یک حلقه بازگشتی هستند که منجر میشود اطلاعاتی که از لحظات قبلی بدست آمده از بین نرود و در شبکه باقی بماند.



مشکلی به نام وابستگی‌های بلندمدت

یکی از جذابیت های شبکه‌های عصبی بازگشتی این است که آن ها ممکن است بتوانند اطلاعات که قبلاً مشاهده شده را به کاری که در حال حاضر در حال انجام است مرتبط سازد، برای مثال استفاده از فریم های قبلی یک ویدئو میتوان در فهم فریم کنونی کمک گرفت. ممکن است مواردی وجود داشته باشد که ما به اطلاعات بیشتری نیاز داشته باشیم. فرض کنید قصد داریم کلمه بعدی در جمله «من زبان فرانسه را خیلی راحت صحبت می‌کنم... زادگاه من کشورِ .....» با توجه به اطلاعات اخیر (یعنی چهار پنج کلمه قبل از آخرین کلمه)، می‌توان گفت که کلمه آخر احتمالا اسم یک کشور است، ولی اگر بخواهیم دقیقاً متوجه بشویم چه کشوری است، ما نیاز داریم به اطلاعات دورتر (یعنی تا ده یا بیست کلمه قبل از آخرین کلمه) دسترسی داشته باشیم. به صورت کلی ممکن است فاصله بین اطلاعات مرتبط و جایی که به این اطلاعات نیاز داریم زیاد باشد. هر چه این فاصله افزایش پیدا میکند، شبکه های عصبی بازگشتی قدرتشان را در به یاد آوردن و استفاده از اطلاعاتی که در گذشته دورتر یاد گرفته‌اند از دست میدهند.

LSTM

هدف از طراحی شبکه هایLSTM ، حل کردن مشکل وابستگی بلندمدت بود. به این نکته مهم توجه کنید که به یاد سپاری اطلاعات برای بازه‌های زمانی بلند مدت، رفتار پیش‌فرض و عادی شبکه‌های LSTM است و ساختار آن‌ها به صورتی است که اطلاعات خیلی دور را به خوبی یاد میگیرند که این ویژگی در ساختار آن‌ها نهفته است.

همه شبکه های عصبی بازگشتی به شکل دنباله ای (زنجیره‌ای) تکرار شونده از ماژول های (واحد های) شبکه های عصبی هستند. در شبکه های عصبی بازگشتی استاندارد، این ماژول های تکرار شونده ساختار ساده ای دارند، برای مثال تنها شامل یک لایه تانژانتِ هایپربولیک (tanh)هستند. شبکه های LSTM نیز چنین ساختار دنباله یا زنجیره مانندی دارند ولی ماژولِ تکرار شونده ساختار متفاوتی دارد. به جای داشتن تنها یک لایه شبکه عصبی، 4 لایه دارند که طبق ساختار ویژه ای با یکدیگر در تعامل و ارتباط هستند.