HW5

使用 Pytorch

設計一個合理的方法，某 A 公司隔天預測股價

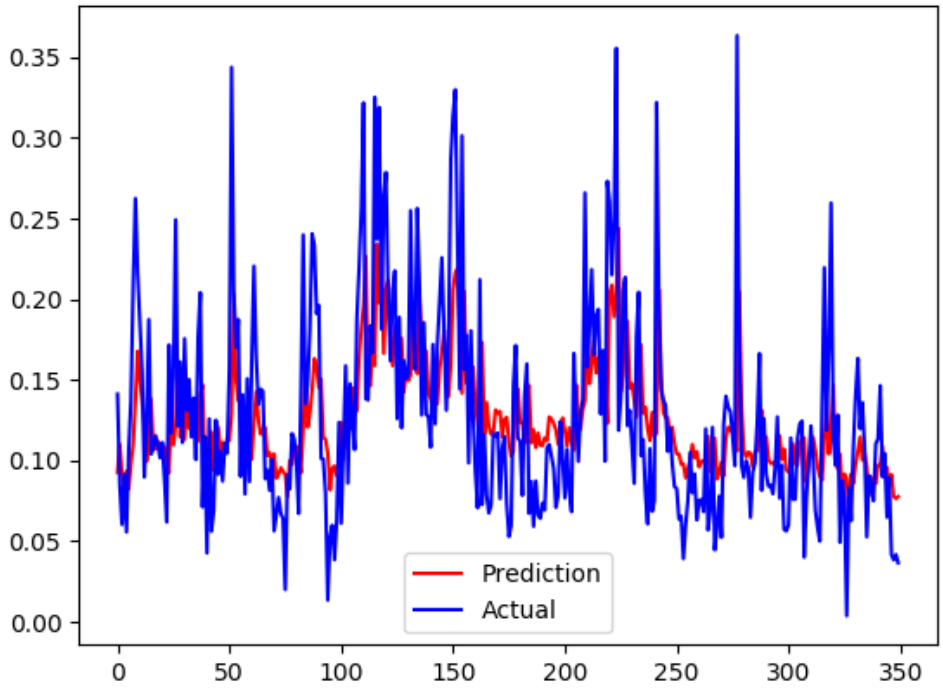
- 使用前 10 天資料做預測
- 80%做訓練 20%做測試
- 均使用 STT 公司

train_test.py preprocess.py ×

```python
from sklearn import preprocessing
import pandas as pd
import numpy as np
def normalize_data(df):
    min_max_scaler = preprocessing.MinMaxScaler()
    df['open'] = min_max_scaler.fit_transform(df.open.values.reshape(-1,1))
    df['close'] = min_max_scaler.fit_transform(df.close.values.reshape(-1,1))
    df['high'] = min_max_scaler.fit_transform(df.high.values.reshape(-1,1))
    df['low'] = min_max_scaler.fit_transform(df.low.values.reshape(-1,1))
    df['volume'] = min_max_scaler.fit_transform(df.volume.values.reshape(-1,1))
    return df

def data_split(stock, seq_len):
    amount_of_features = len(stock.columns) # 5
    data = stock.as_matrix()
    seq_len = 10 # 10 days before
    sequence_length = seq_len + 1 # index starting from 0
    result = []

    for index in range(len(data) - sequence_length): # maxmimum date = lastest date - sequence length
        result.append(data[index: index + sequence_length]) # index : index + 15days

    result = np.array(result)
    row = round(0.80 * result.shape[0]) # 80% split
    train = result[:int(row), :] # 80% date, all features

    x_train = train[:, :-1]
    y_train =np.array(train[:, -1][:,-1])

    x_test = result[int(row):, :-1]
    y_test = np.array(result[int(row):, -1][:,-1])
```

直接用助教提供的模型

畫出訓練loss與測試accuracy curves



以測試資料每一天為單位:

data = stock.as_matrix()
epoch 0, loss 5.130434510647319e-05
epoch 1, loss 2.0575862436089665e-05
epoch 2, loss 1.0688127076718956e-05
epoch 3, loss 6.127215328888269e-06
epoch 4, loss 3.8885091271367855e-06
epoch 5, loss 2.573288384155603e-06
epoch 6, loss 1.702827944427554e-06
epoch 7, loss 1.1196422065040679e-06
epoch 8, loss 7.446545851053088e-07
epoch 9, loss 5.102380669086415e-07
epoch 10, loss 3.6380410506353655e-07
epoch 11, loss 2.7090220555692213e-07
epoch 12, loss 2.091930326741931e-07
epoch 13, loss 1.6542701075650257e-07
epoch 14, loss 1.3246750540929497e-07

epoch 15, loss 1.0657749527354099e-07
epoch 16, loss 8.592413536234744e-08
epoch 17, loss 6.952112840963309e-08
epoch 18, loss 5.656664470166106e-08
epoch 19, loss 4.631214522987648e-08
epoch 20, loss 3.8139987168506195e-08
epoch 21, loss 3.1655478238690193e-08
epoch 22, loss 2.644408425567235e-08
epoch 23, loss 2.2217784945155472e-08
epoch 24, loss 1.8746900920518783e-08
epoch 25, loss 1.582453279524998e-08
epoch 26, loss 1.3309034763153704e-08
epoch 27, loss 1.1124009269281032e-08
epoch 28, loss 9.204700113230047e-09
epoch 29, loss 7.540580604370462e-09
epoch 30, loss 6.095648304597325e-09
epoch 31, loss 4.863309754910006e-09
epoch 32, loss 3.7965683929996885e-09
epoch 33, loss 2.8937074958435005e-09
epoch 34, loss 2.1187320786708597e-09
epoch 35, loss 1.4745824561401832e-09
epoch 36, loss 9.52818268729061e-10
epoch 37, loss 5.501101796312469e-10
epoch 38, loss 2.594726655757995e-10
epoch 39, loss 7.980288652120748e-11
epoch 40, loss 3.469446951953614e-12
epoch 41, loss 2.6047219936486954e-11
epoch 42, loss 1.42464040564505e-10
epoch 43, loss 3.4638980572765377e-10
epoch 44, loss 6.311831057814743e-10
epoch 45, loss 9.94194948589211e-10
epoch 46, loss 1.4257830471819943e-09
epoch 47, loss 1.911439007074023e-09
epoch 48, loss 2.440834201422603 7e-09
epoch 49, loss 3.01028624249966e-09

使用 Tensorflow

設計一個合理的方法，某 A 公司隔天預測股價

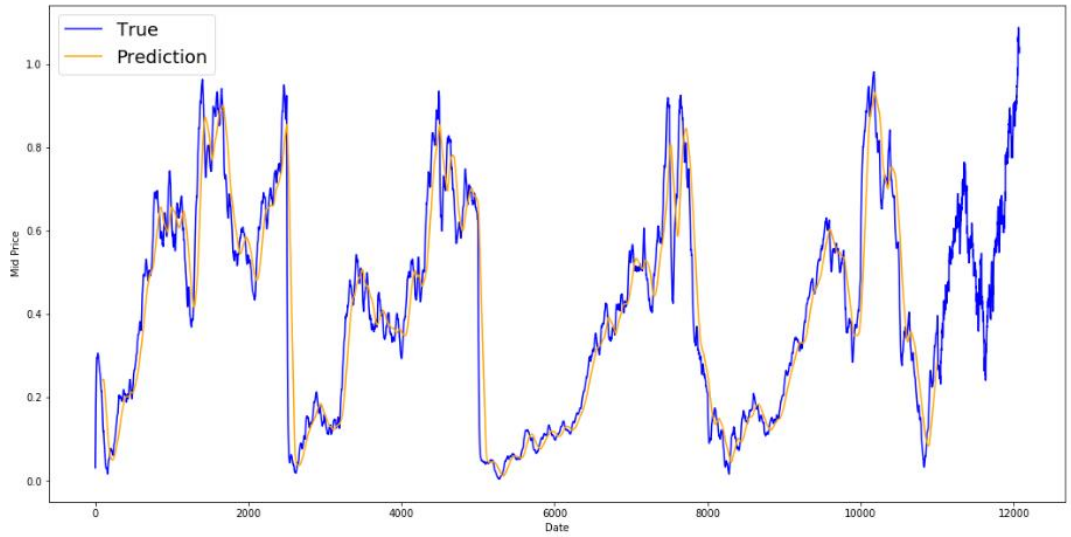
- 使用 1970-2017 的 hp 美股資料
- 使用前 11000 筆作為訓練資料；11000 為測試資料

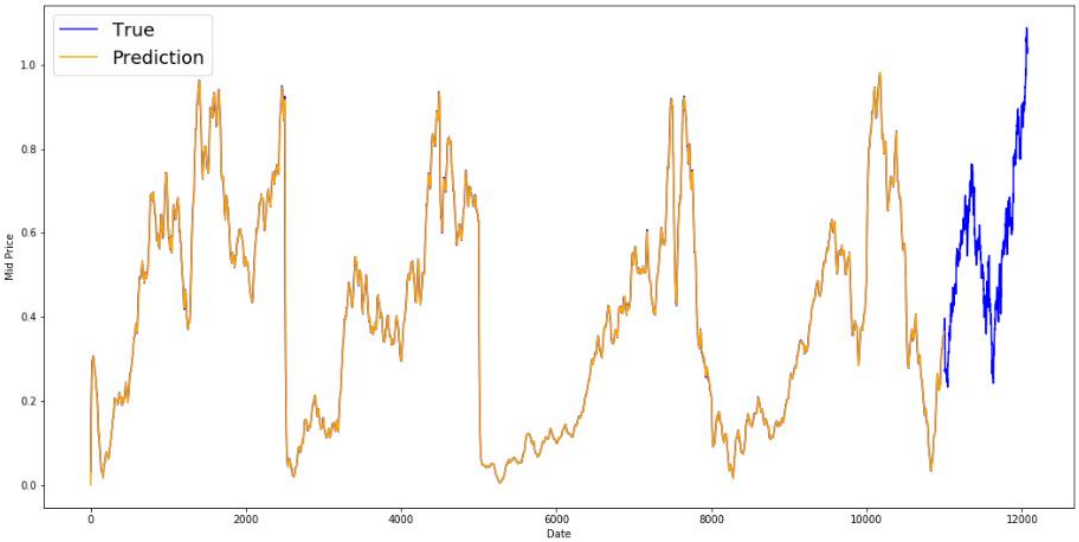
```
# plt.figure(figsize = (18,9))
# plt.plot(range(df.shape[0]),(df['Low']+df['High'])/2.0)
# plt.xticks(range(0,df.shape[0],500),df['Date'].loc[::500],rotation=45)
# plt.xlabel('Date',fontsize=18)
# plt.ylabel('Mid Price',fontsize=18)
# plt.show()

# First calculate the mid prices from the highest and lowest
high_prices = df.loc[:,'High'].as_matrix()
low_prices = df.loc[:,'Low'].as_matrix()
mid_prices = (high_prices+low_prices)/2.0

train_data = mid_prices[:11000]
test_data = mid_prices[11000:]

# Scale the data to be between 0 and 1
# When scaling remember! You normalize both test and train data with respect to training data
# Because you are not supposed to have access to test data
scaler = MinMaxScaler()
train_data = train_data.reshape(-1,1)
test_data = test_data.reshape(-1,1)

# Train the Scaler with training data and smooth data
smoothing_window_size = 2500
for di in range(0,10000,smoothing_window_size):
    scaler.fit(train_data[di:di+smoothing_window_size,:])
    train_data[di:di+smoothing_window_size,:] = scaler.transform(train_data[di:di+smoothing_window_size,:])
```

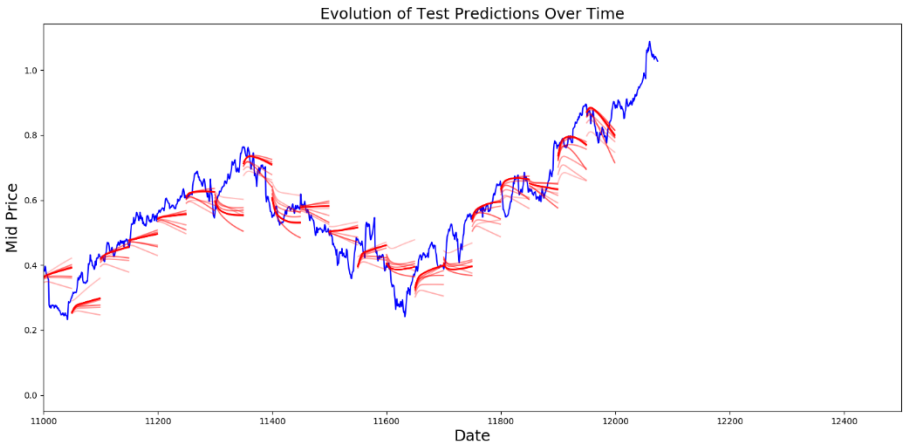
使用網路上找尋的模型

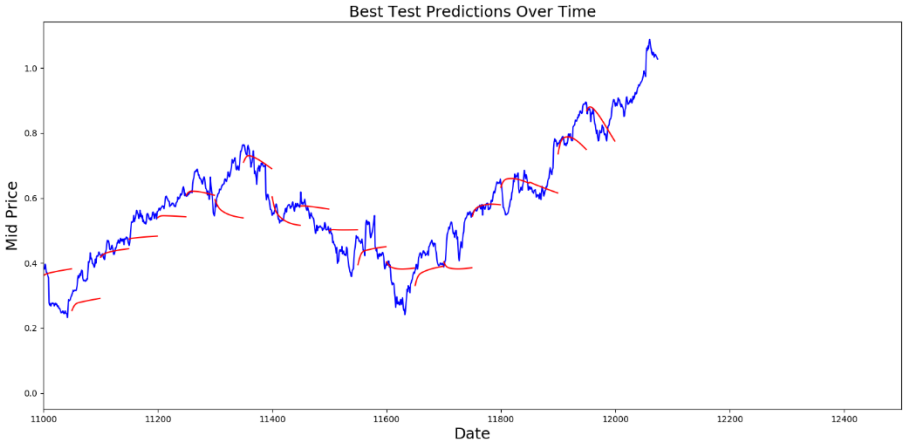
參考資料：https://www.datacamp.com/community/tutorials/lstm-python-stock-market#download

畫出訓練loss與測試accuracy curves


True
Prediction
Mid Price
Date
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
10000
12000


True
Prediction
Mid Price
Date
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
10000
12000





以測試資料每一天為單位:

Initialized
Average loss at step 1: 2.154257
    Test MSE: 0.02947
    Finished Predictions
Average loss at step 2: 0.172901
    Test MSE: 0.00854
    Finished Predictions
Average loss at step 3: 0.083486
    Test MSE: 0.00302
    Finished Predictions
Average loss at step 4: 0.066582

Test MSE: 0.00282
Finished Predictions
Average loss at step 5: 0.055656
Test MSE: 0.00283
Finished Predictions
Average loss at step 6: 0.050306
Test MSE: 0.00249
Finished Predictions
Average loss at step 7: 0.050464
Test MSE: 0.00234
Finished Predictions
Average loss at step 8: 0.049671
Test MSE: 0.00261
Finished Predictions
Average loss at step 9: 0.044614
Test MSE: 0.00261
Finished Predictions
Average loss at step 10: 0.044739
Decreasing learning rate by 0.5
Test MSE: 0.00235
Finished Predictions
Average loss at step 11: 0.041481
Test MSE: 0.00236
Finished Predictions
Average loss at step 12: 0.038258
Test MSE: 0.00264
Finished Predictions
Average loss at step 13: 0.037451
Decreasing learning rate by 0.5
Test MSE: 0.00256
Finished Predictions
Average loss at step 14: 0.036881
Test MSE: 0.00241
Finished Predictions
Average loss at step 15: 0.035269
Test MSE: 0.00252
Finished Predictions
Average loss at step 16: 0.037331

Decreasing learning rate by 0.5
Test MSE: 0.00238
Finished Predictions
Average loss at step 17: 0.041072
Test MSE: 0.00252
Finished Predictions
Average loss at step 18: 0.036354
Test MSE: 0.00258
Finished Predictions
Average loss at step 19: 0.037930
Decreasing learning rate by 0.5
Test MSE: 0.00247
Finished Predictions
Average loss at step 20: 0.037313
Test MSE: 0.00254
Finished Predictions
Average loss at step 21: 0.038118
Test MSE: 0.00245
Finished Predictions
Average loss at step 22: 0.037330
Decreasing learning rate by 0.5
Test MSE: 0.00243
Finished Predictions
Average loss at step 23: 0.035483
Test MSE: 0.00239
Finished Predictions
Average loss at step 24: 0.037280
Test MSE: 0.00242
Finished Predictions
Average loss at step 25: 0.035315
Decreasing learning rate by 0.5
Test MSE: 0.00241
Finished Predictions
Average loss at step 26: 0.036480
Test MSE: 0.00243
Finished Predictions
Average loss at step 27: 0.037379
Test MSE: 0.00241

Finished Predictions
Average loss at step 28: 0.036528
Decreasing learning rate by 0.5
Test MSE: 0.00240
Finished Predictions
Average loss at step 29: 0.038035
Test MSE: 0.00240
Finished Predictions
Average loss at step 30: 0.036815
Test MSE: 0.00247
Finished Predictions