Prediction of TLT price with LSTM

Gan Luan   Wenlong Feng

Summary

Predicting the stock price is always of great interest for financial business. Deep learning techniques are proved to be useful in many areas. In this project, we tried to use deep learning methods to predict the TLT price. The model used is LSTM, a effective recurrent neural network for time series data. In this report, we introduced how the original data were preprocessed for the model. Then the model architecture was described together with the hyperparameters. Model performance was then evaluated in two different time ranges. Limitations of the project and future improvement direction were discussed in the end.

1. Data Preprocessing

The raw TLT daily data contain six features (date, open price, close price, high price, low price, and volume). We used the later five to generate new features and discarded the date attribute, since it is not easy to be directly used in the LSTM model. Instead, we derived weekday and month information from date data as two input features. In the account of potential long-term effects on bond market price, we summarized averaged open, close, high, and low price during prior 90 trading days, prior 180 trading days, and prior 360 trading days, respectively. These averaged price data were appended to the raw daily TLT data as new input features. In order to keep the data consistency, the first 360 records of the raw data were only used to calculate long-term price features for the subsequence records, rather than input into our neural network. Therefore, each record in the input data contains 19 features, tab 1.

Table 1 Features of input data

No.	Feature Name	Description
1	Open	Daily open price
2	High	Daily high price
3	Low	Daily low price

4	Close	Daily close price
5	Volume	Daily trade volume
6	Week	The day of the week (0~4)
7	Month	The month of the year (0~11)
8	90_Op	Average open price in prior 90 trading days
9	90_Hi	Average high price in prior 90 trading days
10	90_Lw	Average low price in prior 90 trading days
11	90_Cl	Average close price in prior 90 trading days
12	180_Op	Average open price in prior 180 trading days
13	180_Hi	Average high price in prior 180trading days
14	180_Lw	Average low price in prior 180 trading days
15	180_Cl	Average close price in prior 180 trading days
16	360_Op	Average open price in prior 360 trading days
17	360_Hi	Average high price in prior 360 trading days
18	360_Lw	Average low price in prior 360 trading days
19	360_Cl	Average close price in prior 360 trading days

Subsequence to the generation of new input features, we normalized data feature by feature except for “Week” and “Month” using the MinMaxScaler from Sci-kit learn. The parameters of data normalization were recorded for normalizing further input and recovering model output.

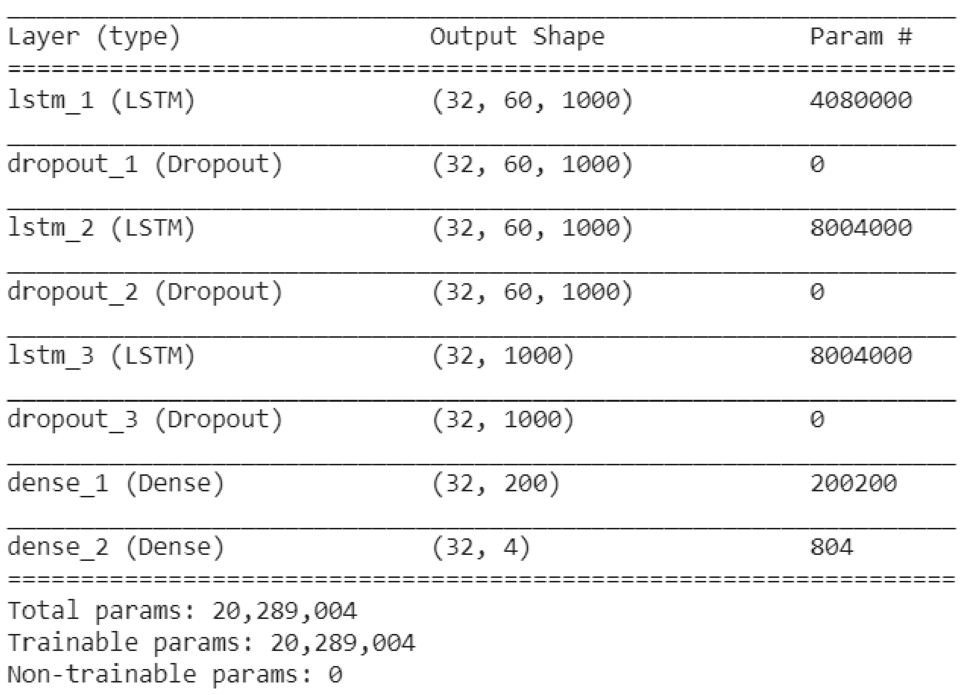
The following process of normalization was rearranging the time sequence of the input data. We define two hyperparameters in this process which are “TIME_STEPS” and “BATCH_SIZE.” The former controls the number of records back in time that used to predict the market prices in the next day, and the latter is a parameter in the step of trimming model input data. We assigned ‘TIME_STEPS =60,’ that is, we intended to use prior 60-day data to predict bond market prices on the 61st day. We created a 3-D matrix “x_total” to store the prior 60-day data for each day and a 2-D matrix “y_total” to store the corresponding next day market price (open, high, low, and close) for the prior 60-day data. The first dimension of x_total matched with the first dimension of y_total. The second dimension of x_total equals to “TIME_STEP,” and the third dimension of x_total equals to features. Subsequently, we randomly shuffled the sequence in the first dimension of x_total and y_total, and their first dimensions were still matched. Then we separated the x_total and y_total into three parts: training data (80%), validation data (10%), and test data (10%). At last, we trimmed the data volumes of these subsets to ensure that the size in their first dimension divided by BATCH_SIZE is an integer. In our process, BATCH_SIZE was assigned to 32.

2. LSTM and Model Architecture

Predicting stock price is a typical time series problem. Recurrent Neural Networks (RNN) are one of the best neural network to for these problems. Currently the most effective sequence models used in practical applications are called gated RNNs. Long short-term memory (LSTM), which is used in this project, is one of the gated RNNs. The key to LSTM is the cell state, which has an internal recurrence in addition to the outer recurrence of the RNN. LSTM has the ability to remove or add information to the cell state, carefully regulated by structures called gates. Three gates are used for LSTM. Forget gate controls the self-loop weight. External input gate and output gate control the amount of information taken from input and information produced from output. By controlling the information flow with gates, LSTM is capable of learning long-term dependencies.

Our recurrent neural network has five layers of neurons. The architecture is shown in Table2. The first three layers are three LSTM layers. Each LSTM layer has 1000 neurons and conducts the dropout regularization with dropout rate as 0.2. The shape of input for the first LSTM layer is as (BATCH_SIZE, TIME_STEPS, Feature size). The subsequence LSTM layers use their closest previous layer output as their inputs. The fourth layer and output layer are feed-forward fully connected layers. The fourth layer has 200 neurons and uses ‘relu’ as the activation function. The output layer has 4 neurons and uses a sigmoid function as its activation. The optimizer adopted in the model is “RMSprop” with a learning rate as 0.001. We adopted “mean_squared_error” as the loss function.

Table 2. RNN architectures



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (32, 60, 1000)            4080000   
_________________________________________________________________
dropout_1 (Dropout)          (32, 60, 1000)            0         
_________________________________________________________________
lstm_2 (LSTM)                (32, 60, 1000)            8004000   
_________________________________________________________________
dropout_2 (Dropout)          (32, 60, 1000)            0         
_________________________________________________________________
lstm_3 (LSTM)                (32, 1000)                8004000   
_________________________________________________________________
dropout_3 (Dropout)          (32, 1000)                0         
_________________________________________________________________
dense_1 (Dense)              (32, 200)                 200200    
_________________________________________________________________
dense_2 (Dense)              (32, 4)                   804       
=================================================================
Total params: 20,289,004
Trainable params: 20,289,004
Non-trainable params: 0
```

During the model training, we assigned the batch size in the model fitting as 32 and intended to run 200 epochs. However, the earlystopping regularization ceased the training at the 87th epoch. The patience of the earlystopping regularization is 15. The training loss and validation loss is displayed in figure 1.

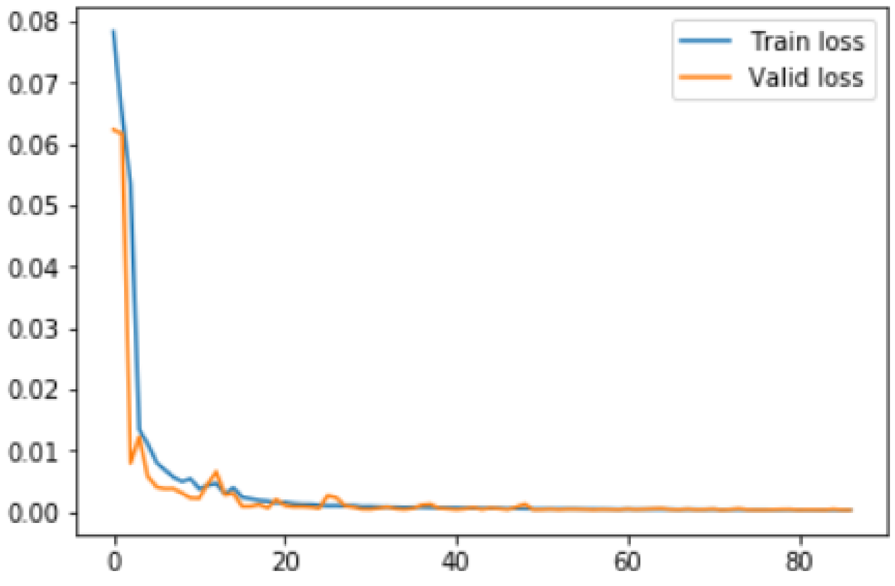


Figure 1 Training loss v.s. Validation loss

3. Output Analysis

To evaluate the performance of the model, TLT prices (Open, High, Low, and Close) of the past 3200 days (up to May, 14) were predicted and compared with the real data. Mean square error of the predicted data and the real data is 0.905. Square root of this error is 0.95, which is only about 0.9% compared with the scale of the TLT price (around $110 for past 3200 days). This suggest that in general our model performs well. Then the predicted and real TLT price for each category (open, high, low, and close) were also plotted to further evaluate the model performance. Basically the plot are very similar for all these four categories. The plot of open prices are shown below (see Fig. 2). It is clear that the prediction can closely follow the trend of the real price with some difference at certain points. Most of these differences occur at the turning points where the trend of the price changes either from increase to decrease or otherwise. It is not unexpected, since the prediction always has a lag in predicting the trend of the price. However, correctly predicting these turning points are critical in real financial business. Thus a further direction to improve the prediction in the turning point.

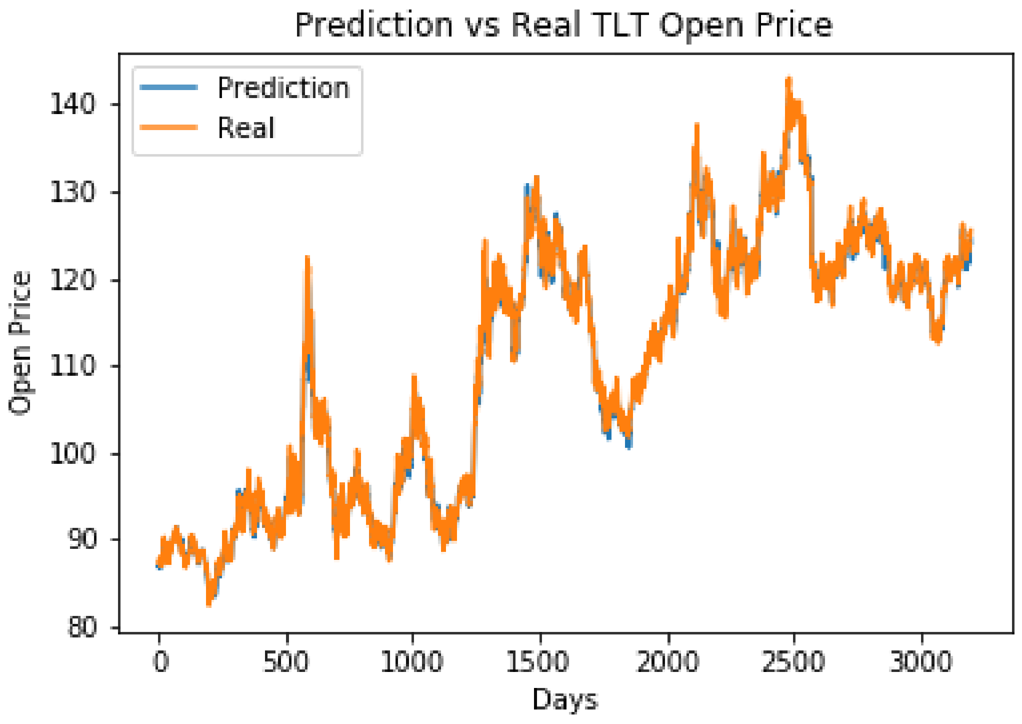


Figure 2 Prediction vs Real TLT Open Price for past 3200 Days

To take a closer look, the predicted and real open price for the past 128 days were also plotted (Fig. 3). It also shown that most of the difference appears around the turning point. Basically, that the prediction was very close to the real price for the first around 100 days. And there is a lag for the pattern in real data to be shown in prediction data. As explained above, this lag might be the reason for the big difference at turning points. However, for the most recent 20 days, a bigger difference appears between the predict and real prices. This is a sign of overfitting. The training data set only include data up to April, 16, 2019. No data are provided for model training after that. The model has never seen the recent data. One way to solve this is to modify the model to reduce the overfitting. Another way is to always train the model with the most updated data available.

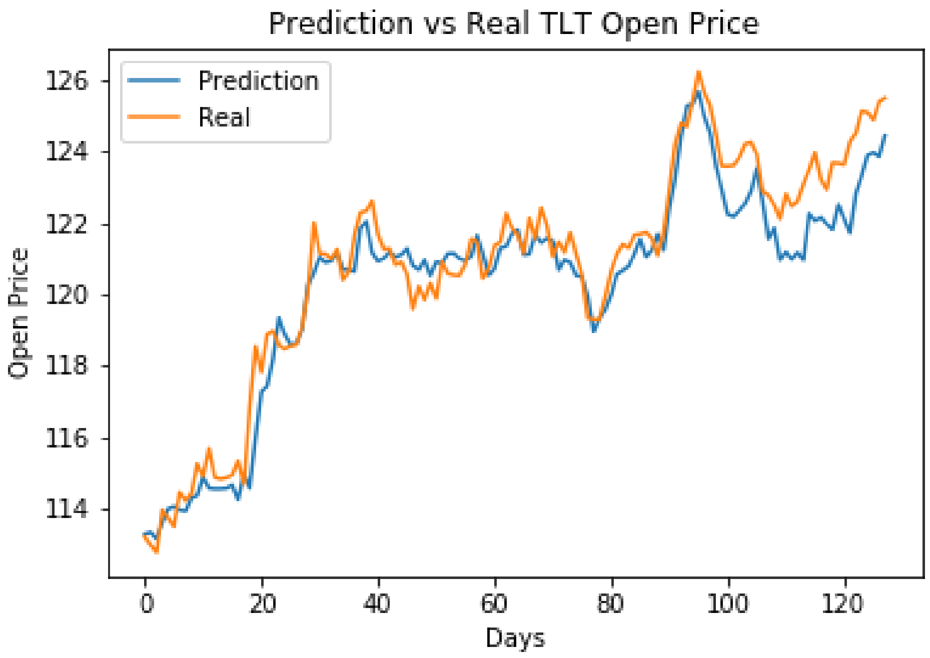


Figure 3 Prediction vs Real Open TLT Price for past 128 Days

4. Limitations and Future Directions

There are several limitations about our models. One is that it does not predict well around the turning points. To solve this problem, we may need to find more features for the model. Basically, we need to find the features that might be the intrinsic reason for the change of the TLT price. Possible features could include the performance of market, the unemployed rate, the bank’s rate, and so on. Finding the related features and preprocess these features are called feature engineering. In real data science application, feature engineering can take most of the time of one project.

Another is that it does not predict the recent result well as mentioned above. One possible reason for this is overfitting. We can try to solve the overfitting problem by using a smaller patience, a larger dropout, or a less complicated network. If lack of recent data rather than overfitting is the reason for this problem, the possible solution is to feed recent data to the model and updated the model regularly. Currently if we want to update the model with new data, we need to add new data to the previous dataset to generate a new complete dataset and then run the model all over again with the new dataset. Clearly this is time consuming. We may modify our model so that it can take the new data without running over the previous data all over again.

Another potential limitation is that we use sigmoid activation function for our final result. This is reasonable since the scaled label are between 0 and 1 and the output of sigmoid function is also between 0 and 1. The limitation is that it will not generate the output that larger than the maximum value or smaller than the minimum value in training set. It will not be a serious problem for this specific project, since recent TLT prices are not close to either the minimum or maximum of the training set. However, it is a potential problem in the furture. One way to solve this is to use a linear function instead of sigmoid function.

We also noticed one interesting thing during our project. We first trained and saved our model. Then we loaded the model and made prediction for the next day. Once we happened to run the prediction code twice and realized that the prediction result changed without any other modifications on the model or on the codes. This is not a unique phenomena with that specific data. This happens to all the data we tried to predict. **Fig. 4** is a plot of predicted open TLT price for May, 14th after running the prediction code for different times. Basically, there is a big difference between the first and second run and then the result tends to be stable with very small change. The result we uploaded are the stable result and the stable result are closer to the real data. However, it is not clear why the result will change after running the same code for different times. This requires further investigation.

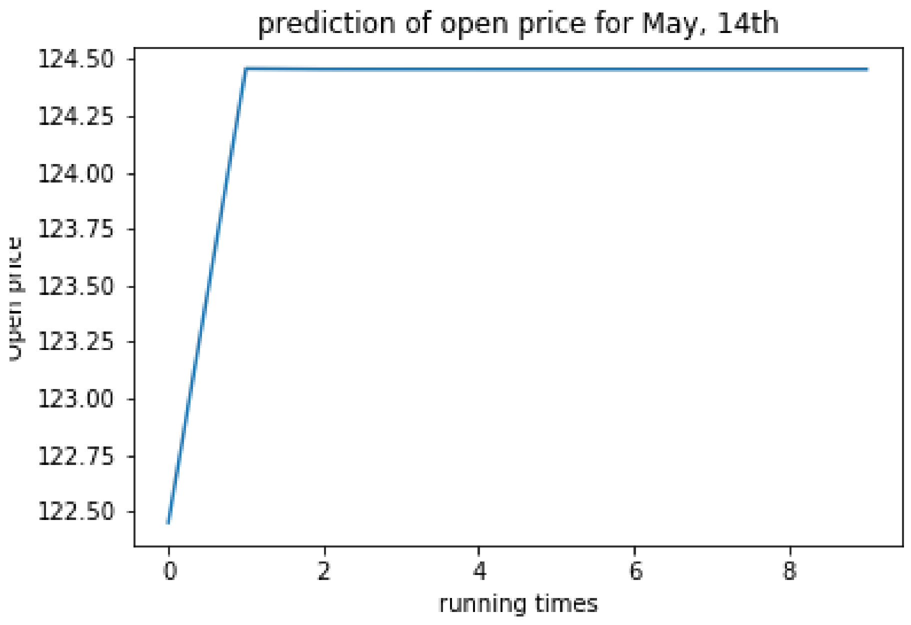


Figure 4 Prediction of open price for May, 14th