XGBoost & LSTM

1 Why using these two methods?

To solve this problem, there are many potential ways. First choice is through ensemble learning, because for most the complex problem, ensemble way always has the best average performance. XGBoost, shortened form of Extreme Gradient Boosting, which is a very popular method in data mining and machine learning competition and engineering works with advantages on non-sensitivity of inputs and high accuracy.

The reason of using LSTM is because LSTM is designed to deal with long term and short term memory, and has pretty good performance when applied on time series problems.

2. implementation

Based on the principle that not to make transaction cost too high, but still need to maximize the value of decision, for these two approaches, decision period is made as an hour, and requiring input data from last two hour to make sure get enough information.

For XGBoost

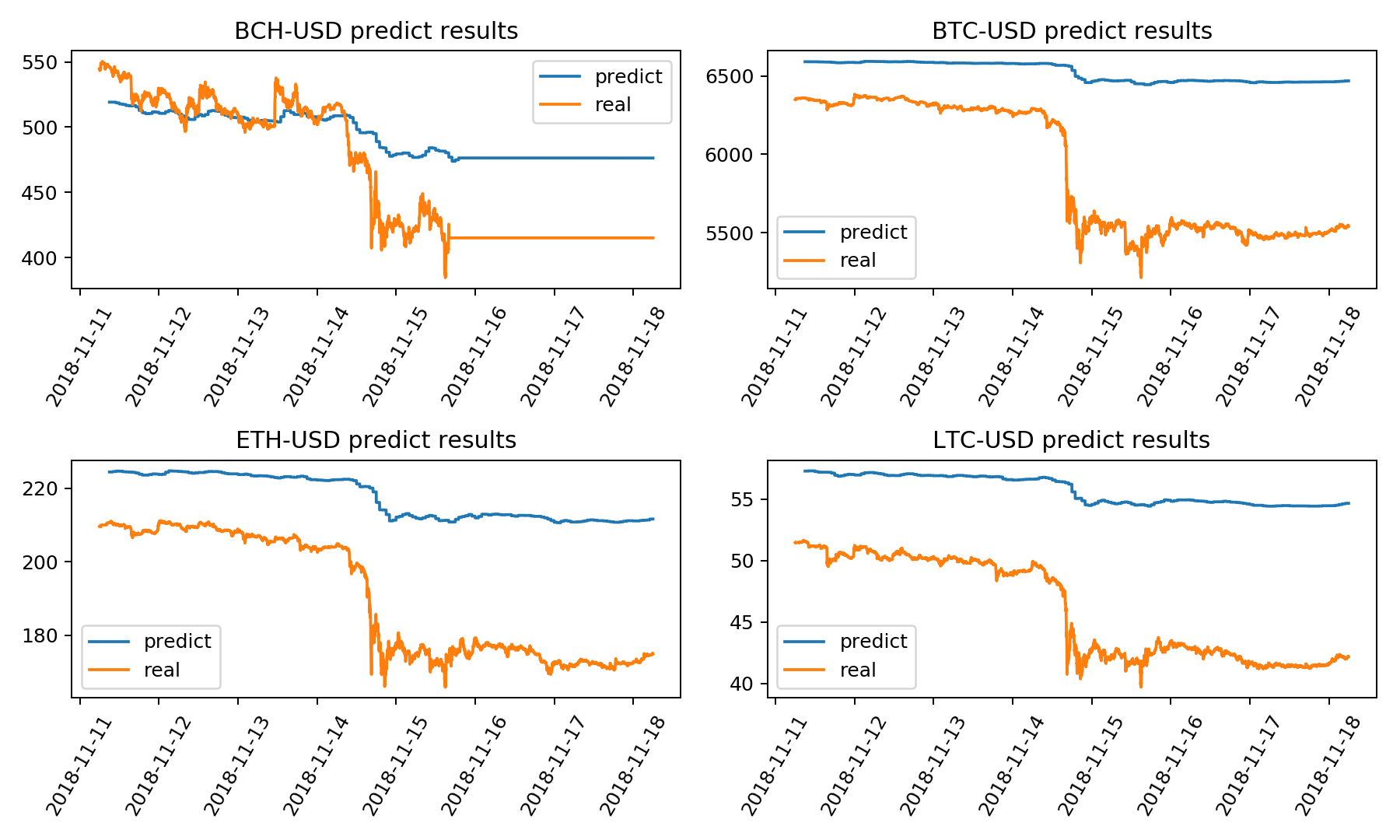
Get idea from back\_test.py, using the average price of open, close, high and low value of each time point to present the price of currencies. In this way, data complexity can get simplified and pressure on model training also get reduced.

Basically, the former two hours are split into four part, each part contains thirty minutes data of currency, and calculate the mean of average price during that time respectively. Then get four average values to represent the currency price features of the last two hour and use them as the training and model input. For training output, we use the mean of average price of the next hour to present the price change of the coming hour. Maybe this kind of method could contain some error, but comparing to using time point directly, this kind of way has better accuracy and application performance.

For model training part, to improve the total accuracy and reduce the error, we combined three XGBoost models with different hyperparameter.

These three models has the same gradient boost method ‘gblinear’, after comparing ‘gblinear’ and ‘gbtree’, we chose the function with better performance. The difference is that the number of estimators risen from 1000 to 1500 and 2000, while the maximum depth dropped from 7 to 5 and 3 respectively.

Below shows the validation plot of currency curves. It’s very clear that the actual price didn’t fit well, but the trend of change has been simulated with very little lag.

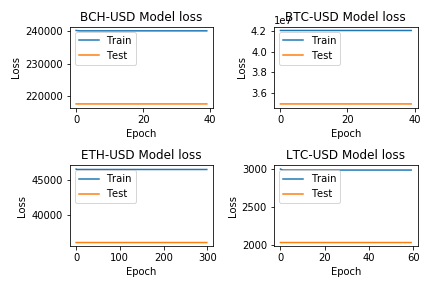


For LSTM

Like the way of data using in XGBoost part, we mainly adopted the average price of currencies to simplify the question and keep consistent with back test file.

The difference is, in this model we adopt all time point data into the model input because of LSTM’s advantage on dealing with high dimension features, and used the last minute price of the next hour as our target.

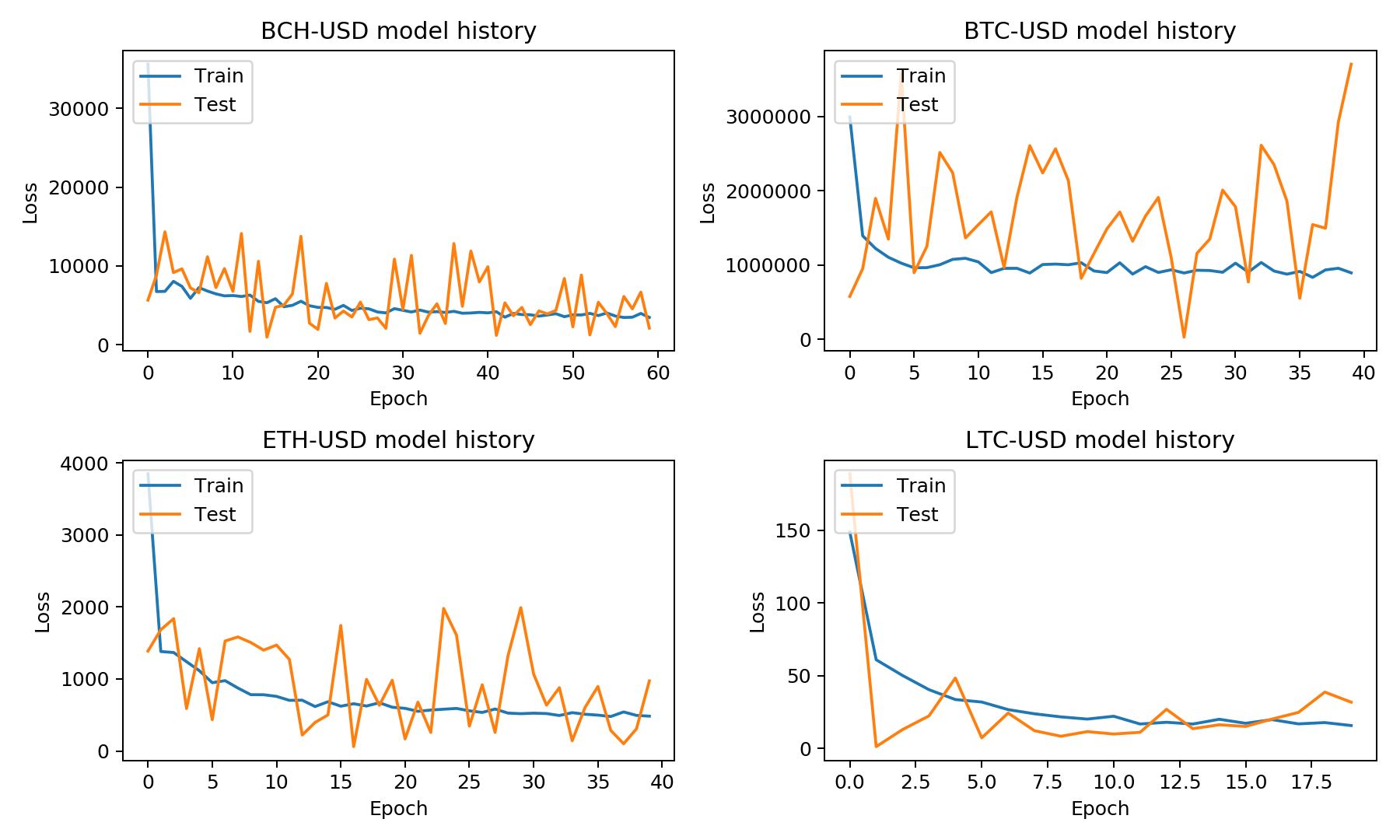
Why not using more data or predicting all time point of the next hour?



The graph above gives the answer. Because LSTM is very sensitive to data, if the format of training input and output are not proper with model’s optimization function and activation function, there will be high probability of under fitting even no fitting.

So, after several times trial, we chose the mentioned feature engineering way and construct our neural network below.

Based on sequential structure, there are three layers of LSTM network with output dimension 64, 128, 64 respectively and one Dence layer with tanh activation function to collect output from former layers and form final output. Between each two layers, we also applied dropout layer with dropout rate 0.2 to avoid overfitting.

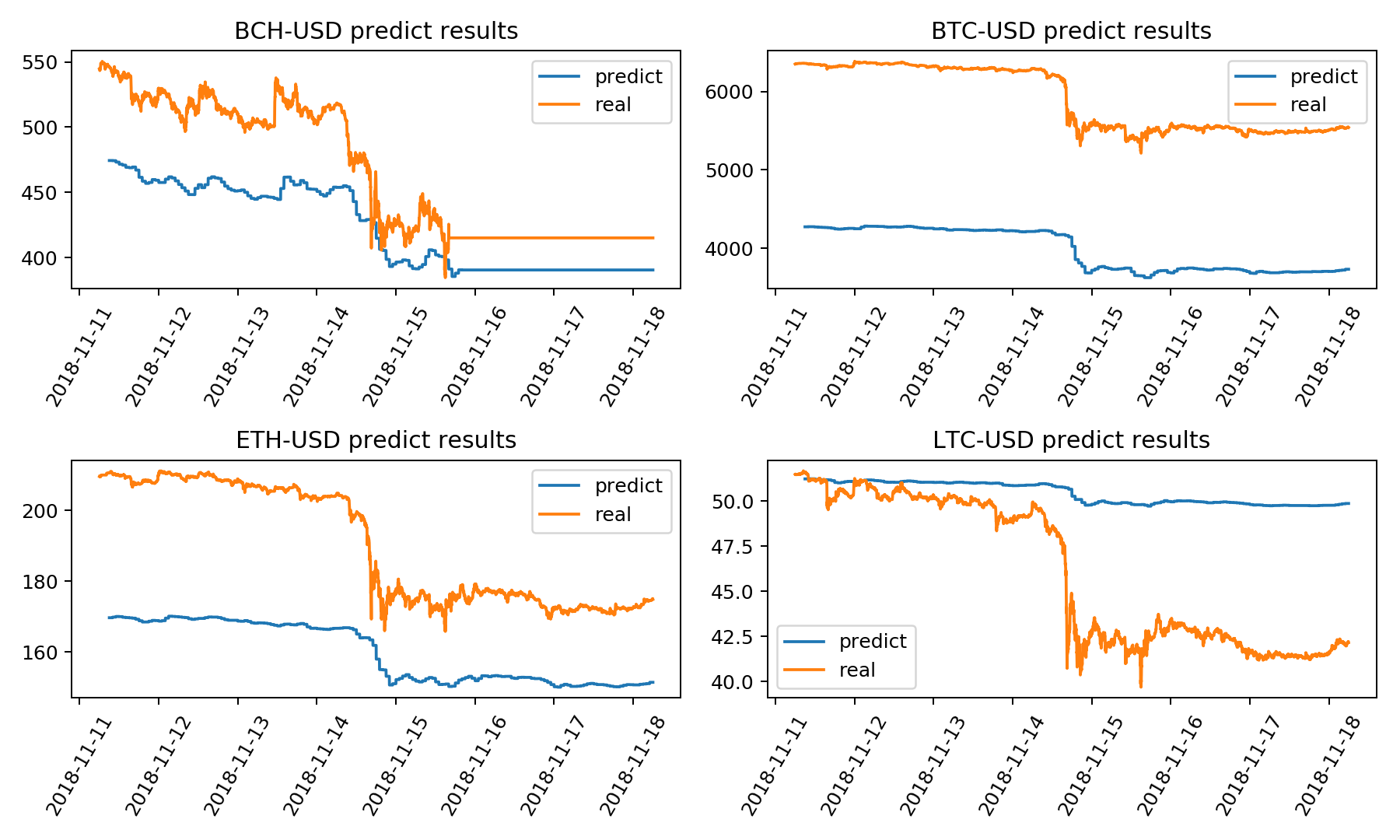


Here is the fitting history.

For currency ETH and LTC, we can find the model can converge well, but model for BCH and BTC could only converge to a certain extend.

Unfortunately, we can see the validation data varies a lot, only on LTC which can get a acceptable curve. However, it’s also reasonable, as a result of high randomness of real world data, the fluctuation of price is influenced by many other factors, it’s indeed hard to use only the price itself to predict the future trend.

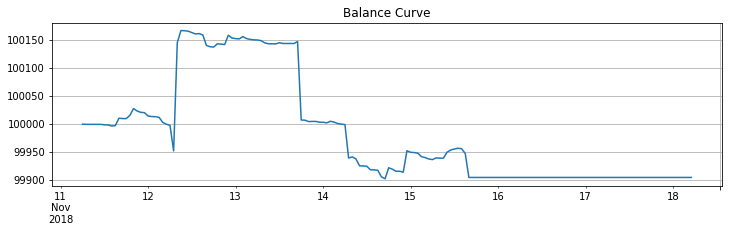
From the predicted data and real data curves of 11-11 to 11-18, we can see LSTM can simulate the trend of change a bit better than XGBoost while the disadvantage on value fitting still remaining. The worse thing is the time lag on LSTM is higher than XGBoost. In this way, maybe LSTM could not have better performance than XGBoost.



We tested these two ways of model construction on past 5 weeks data.

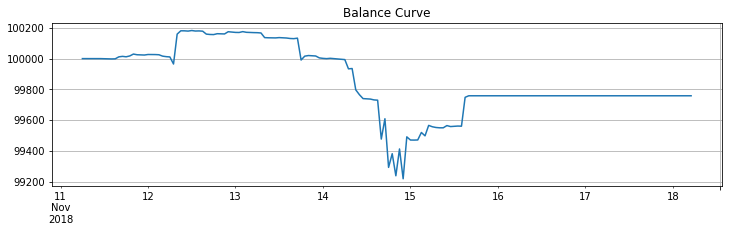
xgboost

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| week | 1014-1021 | 1021-1028 | 1028-1104 | 1104-1111 | 1111-1118 |
| Total return | -0.0017 | -0.0013 | 0.0009 | -0.0057 | -0.0045 |



Lstm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| week | 1014-1021 | 1021-1028 | 1028-1104 | 1104-1111 | 1111-1118 |
| Total return | -0.0034 | -0.0021 | 0.0002 | -0.0069 | -0.0110 |



In total, both kind of model could not have expected return and the total return is easily to be influenced by real market trend.

But we can also conclude that the triple XGBoost combined method has higher positive return and lower loss than LSTM. It also proved the strong ability of ensemble learning and the reason that why XGBoost is widely used to some extends. While we can also see LSTM has a large potential improvement if a more complex neural network can be designed for future works.