Predicting tick's direction of a stock

This report presents the technical details and results for predicting the tick direction of a stock based on its current order book and trade data. The models used for solving this problem include: (1) Linear regression, and (2) Long-Short Term Memory model (LSTM). For both models, they are built based on different selection of features. Some features come from the original data set, while some features are derived from the original ones.

1. Linear Regression

We first construct a linear regression for forecasting the stock movement.

1.1 Some Derived Features

Based on the information provided by the order book, we compute certain features to capture the imbalance between buy and sell orders, which will drive the price to move up or down.

(1) Volume Order Imbalance

The order imbalance is used to classify trades as either buyer or seller initiated. This is done by checking if the trade price was closer to the bid (buy) or ask (sell) of the quoted price. Volume Order Imbalance is defined as follows:

$$\mathrm{VOI}_t = \ \delta V_t^B - \delta V_t^A$$

where

$$\delta V_t^B = \begin{cases} 0, & P_t^B < P_{t-1}^B \\ V_t^B - V_{t-1}^B, & P_t^B = P_{t-1}^B \\ V_t^B, & P_t^B > P_{t-1}^B \end{cases}$$

$$\delta V_t^A = \begin{cases} V_t^A, & P_t^A < P_{t-1}^A \\ V_t^A - V_{t-1}^A, & P_t^A = P_{t-1}^A \\ 0, & P_t^A > P_{t-1}^A \end{cases}$$

Here V_t^B and V_t^A are the bid and ask volumes at time t respectively, and P_t^B and P_t^A are the best bid and ask prices at time t respectively.

(2) Order Imbalance Ratio

We define a new factor---Order Imbalance Ratio (OIR) as:

$$\mathrm{OIR}_t = \frac{V_t^B - V_t^A}{V_t^B + V_t^A}$$

In fact, this factor is given in the order book by, “BIDASKIMBALANCE”. This factor complements the volume order imbalance by allowing us to distinguish cases where the difference is large but the ratio is small.

(3) Mid-Price Basis

Using the traded volume and turnover information, we are able to determine the average trading price between two time-steps. We define the Average Trade Price from [t-1, t] as:

$$TP_t = \begin{cases} M_1, & t = 1 \\ \frac{T_t - T_{t-1}}{V_t - V_{t-1}}, & V_t \neq V_{t-1} \\ TP_{t-1}, & V_t = V_{t-1} \end{cases}$$

where T_t is the turnover (trade amount in money) and V_t is the transaction volume at time t. We define the factor R_t which we call the Mid-Price Basis (MPB) as

$$R_t = TP_t - \frac{M_{t-1} + M_t}{2}$$

where M_t is the mid-price at time t.

(4) Bid-Ask Spread

The bid-ask spread at time t is defined as $S_t = P_t^A$-P_t^B. This information can be used to adjust our regression factors for different levels of liquidity, by dividing them by the instantaneous spread.

1.2 Linear Model

Our strategy uses ordinary least squares to forecast the average mid-price change over the next 20 time steps. The final linear model is presented as the following:

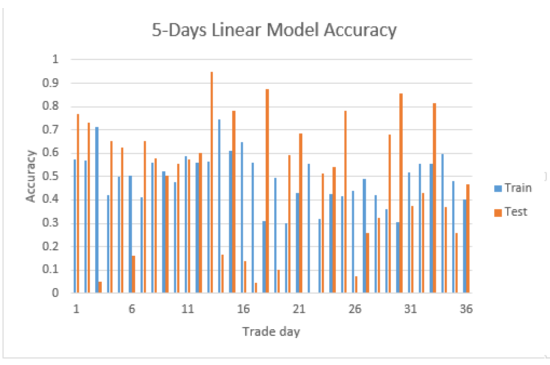
$$\Delta M_{t,20} = \alpha_0 + \sum_{j=0}^{L} \beta_j \frac{VOI_{t-j}}{S_t} + \sum_{j=0}^{L} \gamma_j \frac{OIR_{t-j}}{S_t} + \theta \frac{R_t}{S_t} + \epsilon_t$$

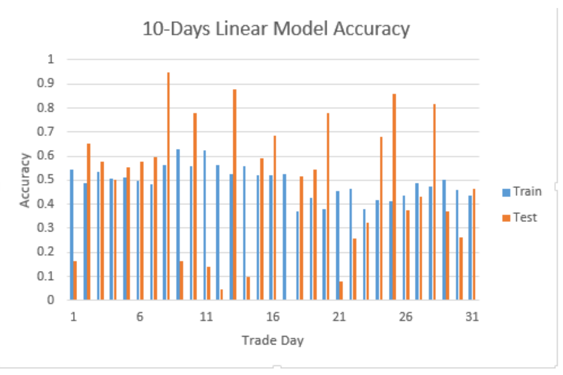
where $\Delta M_{t,20} = \frac{1}{k}\sum_{j=1}^{k} M_{t+j} - M_t$ is the k-step average mid-price change. Also, we let L=5 in this model. We then use the predicted value $\Delta M_{t,20}$ to forecast the tick direction. That is, if $\Delta M_{t,20} > 0.01$ we set the tick direction at time t as UP; if $\Delta M_{t,20} < -0.01$ we set the tick direction at time t as DOWN; otherwise, we let the tick direction as FLAT.

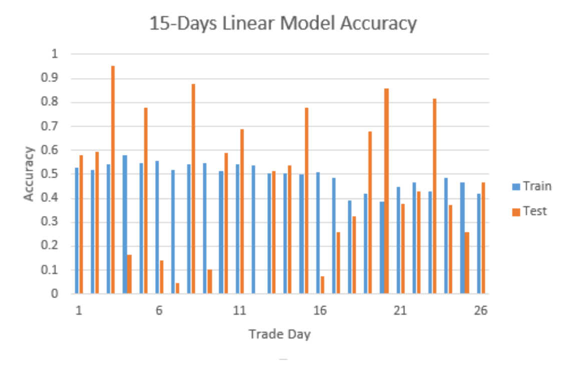
We choose the accuracy to measure the performance of linear models: we count 1 if the prediction matches the actual label or 0 if it is wrong, and then the final count is then divided by the total number of data points.

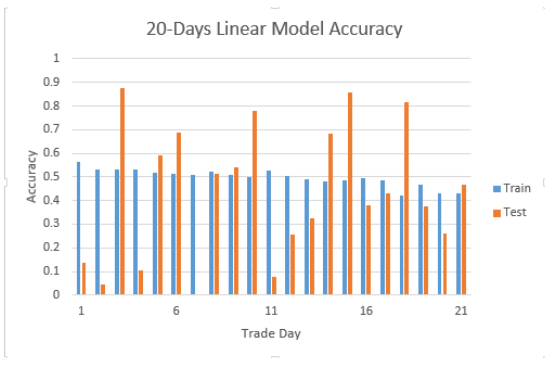
1.3 Results

For the simple linear regression model, we train our model based on 5, 10, 15, and 20 days of trading data in the order book. After that, we use each model to predict the tick direction in the following trading day. The following four figures show our results in the four types of linear models.









As seen above, linear models give a testing accuracy of 0.5-0.6 on average. In some trading days, those linear models perform surprisingly better in the testing set than the training set. However, in some other trading days, those models have low testing accuracy. Linear models are not very robust and predictable based on the trading day. This defect exists no matter whether the trading period increases or not. This is not surprising, considering that linear regression models are more suitable for regression problems, and are usually not adapted for classification problems in supervised learning.

2. Long-Short Term Memory Model

2.1 Basic Setting

(1) Data: Let X and Y denote the features and labels in the training set. Here, we use the "TICKDIR" of the original dataset as label, which is the tick direction we want to forecast. Moreover, we let "UP" and "UPCYCLE" labelled as 1, "FLAT" as 0, and "DOWN" and "DOWNCYCLE" as -1.

(2) RNN Construction: Let n_inputs be the number of features we select in X. Let n_steps be the number of time steps that LSTM will keep in its memory. Let n_hidden_units be the number of neurons in the hidden layer.
RNN has three layers: one input layer, one hidden layer, and one output layer. In the input layer, we let X multiply by a matrix of size n_inputs by n_hidden_units, plus a bias vector. Then, the output of the first layer will pass into the hidden layer, which is a basic LSTM recurrent network cell that has n_hidden_units neurons. In the output layer, we will let the final status of the LSTM recurrent

network cell multiply by a matrix of size n_hidden_units by n_classes. Here, n_classes equals to 3, the 3 values that the tick direction can take.

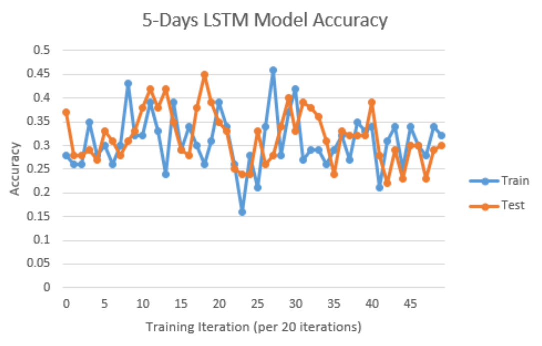
(3) Hyper-Parameters: We set the learning rate to 0.001, the total number of training iterations to 1000, and the batch size to 1000. Without further specification, these hyper-parameters will remain unchanged.

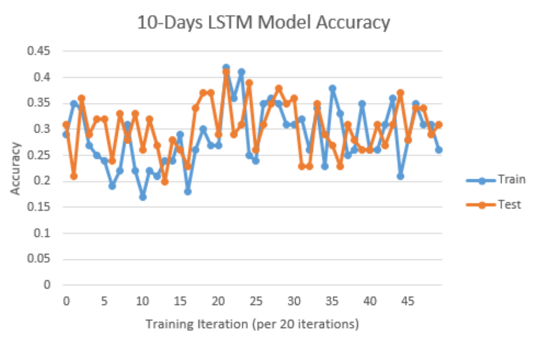
(4) Accuracy: We choose the accuracy to measure prediction's performance. Here, we assume that each of the three classes "FLAT", "UP" and "DOWN" are equipotent.

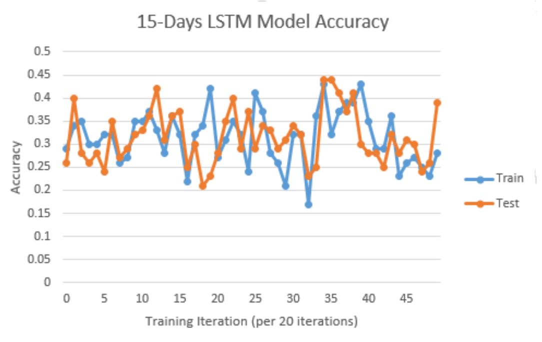
The testing process goes as follow. We set the training range as 5 days, 10 days, 15 days and 20 days, respectively. We start to choose data in the first training range (e.g., 5 days trading data in the order book). This data set will be used to train the model. We measure the accuracy in this training set. On the other hand, we randomly select some data points from ***the first day*** after this training range (i.e., testing data set), and measure the accuracy in this testing data set. After 1000 iterations (hyper-parameter), we move to the second training range, and continue this training-testing process until we reach the last day.

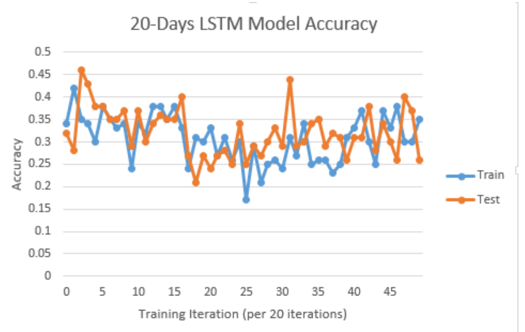
2.2 Small Feature Set

In this section, we consider a small feature set, which consists of only those derived features we considered in the linear regression model.









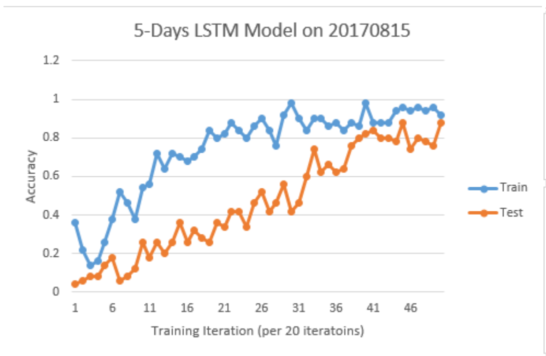
As we can see from the figures above, although we input the same features into the LSTM model, both training and testing accuracies are not higher than what we got with the linear regression model. However, the testing accuracy now becomes more stable. In fact, even when we increase the training iterations and change some other parameters, the accuracy seems not to increase. Thus, we directly turn to use a large feature set as shown in next section.

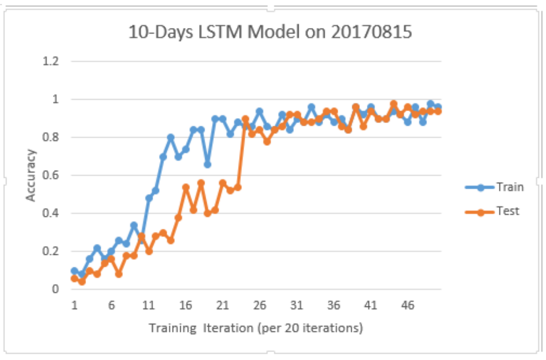
2.3 Large Feature Set

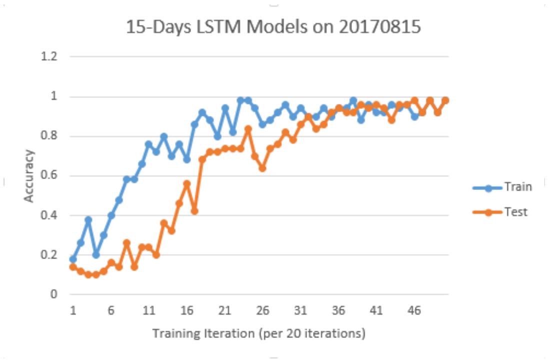
In this section, we consider a large feature set, which contains of all the features given by the original order book, and all derived features we showed before.

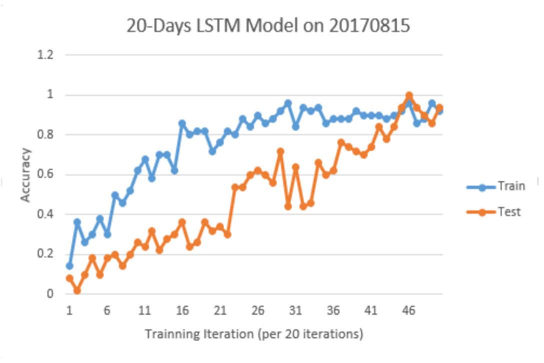
(1) The Effect of Training Range

We first set n_steps = 20 and n_hidden_units = 80, while the training range varies from 5 to 20 days.





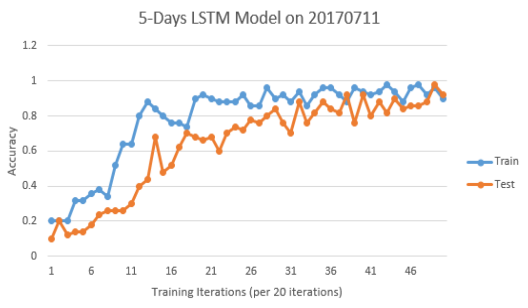


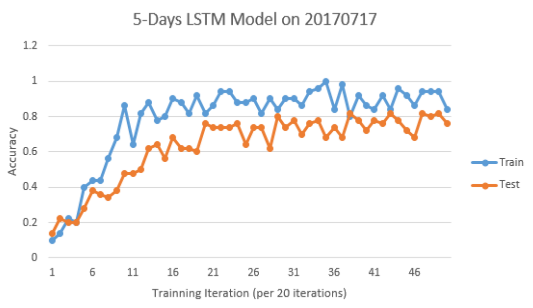


The above four figures show our results with the LSTM model. We train the LSTM model based on 5, 10, 15 and 20 days of trading data, respectively, and then test the model the trade data on 20170815. Here, at each iteration, we randomly select and train some data points from the training set, and test the current model's accuracy in a random batch of testing set (i.e., the trading date on 20170815).

As shown in the above figures, when the training range increases, the model's accuracy in testing set also increases. However, this is not always true, when the training range is too large (e.g. 20 days), the model tends to perform worse than those with smaller training range, as shown in the last figure. The reason is that the trading behaviour in the older history has less effect on the current behaviour, and hence when we include this data in our training set, it acts as noise.

Also, the model we obtained in this way has different performance on different trading data, as shown in the following figures. Though we used the previous 5 days of data to train our model, the accuracy in testing set is higher on 20170711 (above 0.8) and it is lower on 20170717 (below 0.8).





We also present the final training and testing accuracy (after 1000 training iterations) with different training ranges. Here we only show the model performance on the *last 10 trading days* in the order book. The testing accuracy means the accuracy on the target trading day, while the training accuracy means the accuracy on the training set, i.e., the 5, 10, 15, and 20 trading days before target trading day, respectively.

5-Days LSTM Model Accuracy										
Train	0.96	0.92	0.94	0.94	0.84	0.82	0.94	0.94	0.96	0.92
Test	0.98	0.76	0.94	0.94	0.9	0.5	0.94	0.94	0.8	0.98

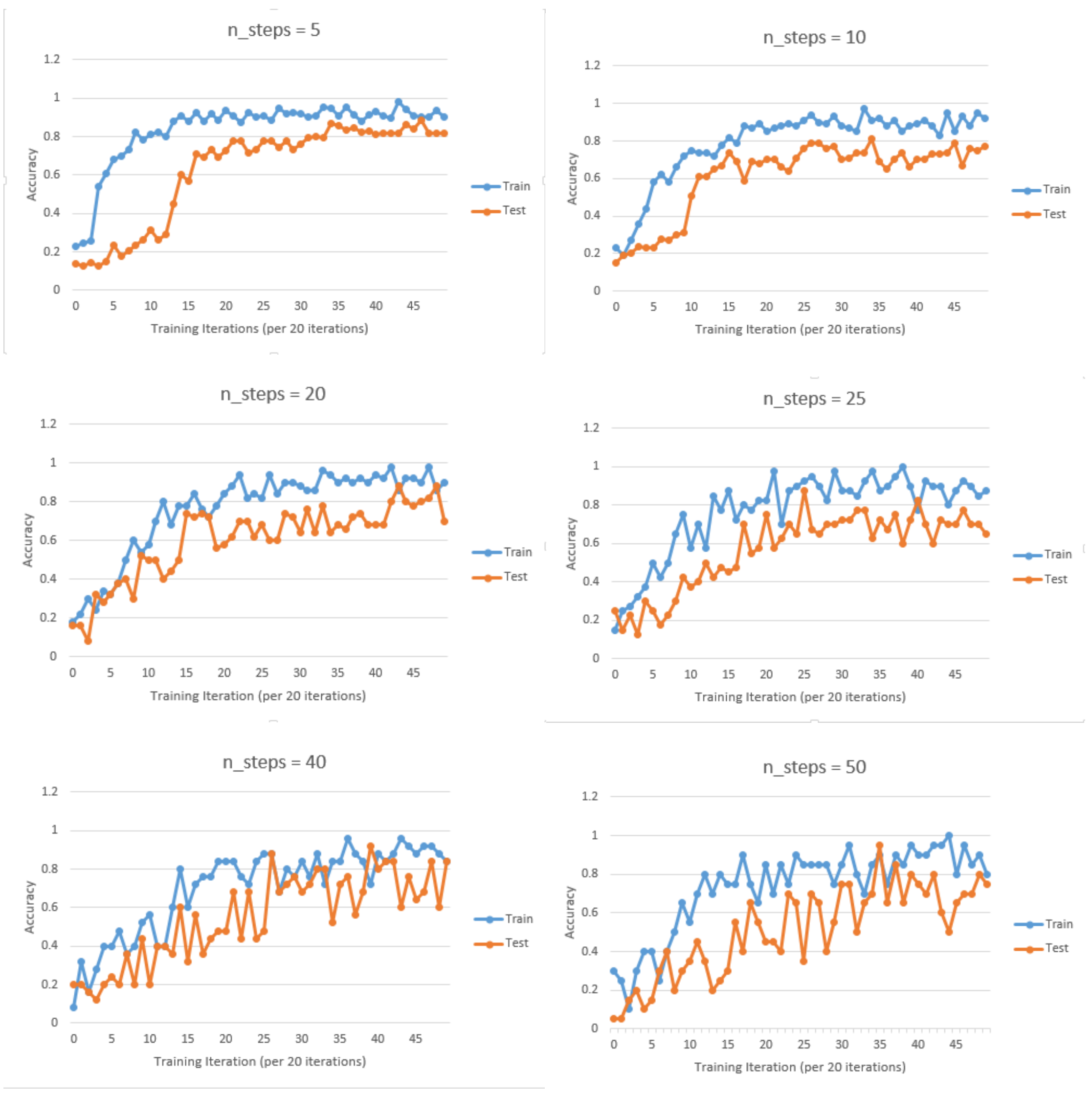
10-Days LSTM Model Accuracy										
Train	0.92	0.94	0.88	0.99	0.98	0.92	0.98	0.96	0.84	0.94
Test	0.94	0.82	0.98	0.88	0.99	0.92	0.98	0.94	0.9	0.99

15-Days LSTM Model Accuracy										
Train	0.98	0.9	0.94	0.9	0.99	0.96	0.94	0.99	0.96	0.92
Test	0.98	0.56	0.94	0.9	0.96	0.94	0.92	0.88	0.88	0.9

20-Days LSTM Model Accuracy										
Train	0.9	0.88	0.9	0.94	0.96	0.88	0.94	0.98	0.92	0.94
Test	0.82	0.64	0.86	0.94	0.94	0.92	0.92	0.9	0.9	0.91

(2) The Effect of Time Steps in LSTM cell

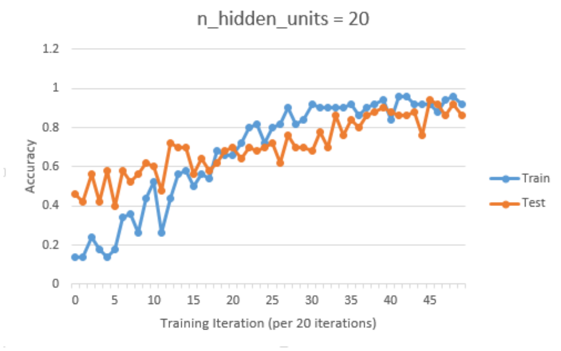
Letting training range = 10 and n_hidden_units = 80, we consider the 10-Days LSTM model with different n_steps. The following figures show the testing accuracy on 20170802, and the training accuracy on the 10-days training set. As we can see from those figures, as the parameter n_steps, namely the time steps in the LSTM model, increases, accuracy decreases and becomes unstable. When n_steps =5, the testing accuracy is almost 0.8; however, when n_steps = 40 or 50, the testing accuracy is almost always below 0.8.

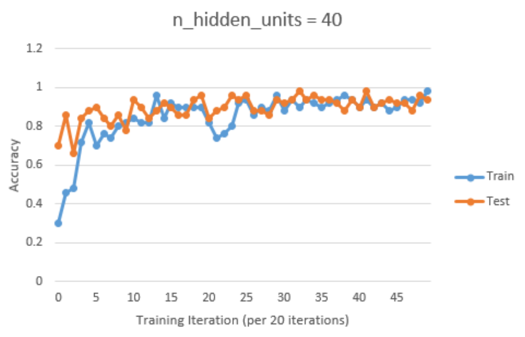


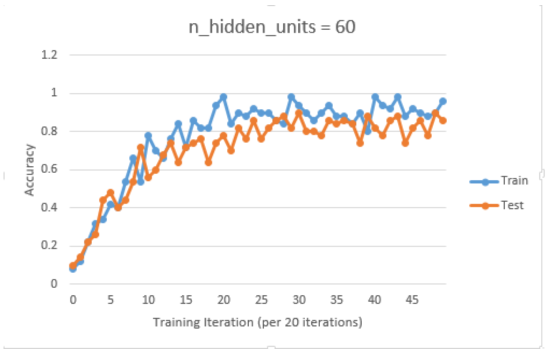
(3) The Effect of Neurons in LSTM cell

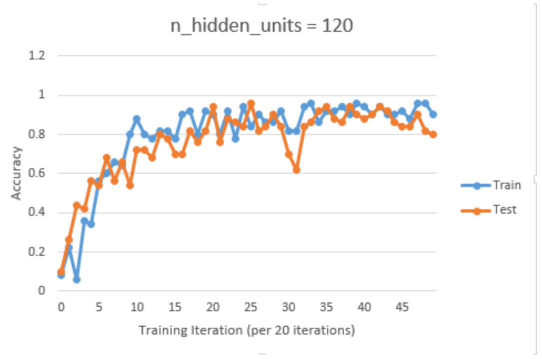
Now we consider the effect of neurons in the LSTM cell. Letting training range = 10 and n_steps = 20, we consider the 10-Days LSTM model with different n_hidden_units. Again, the following figures present the testing accuracy on 20170802, and the training accuracy on the 10-days training set.

Here we can see that, as n_hidden_units, i.e., the number of neurons in the LSTM cell, increases, accuracy increases and then decreases, with the second figure having the highest accuracy among the four. Also, the training accuracy converges to a high level at the fastest speed when n_hidden_units =40.









3. Conclusion

In this problem, we have built two main models: (a) linear regression, and (b) long-short term memory neural network. Both models can capture the motion of tick direction. Besides the features provided by the order book, we also created some new ones to forecast tick direction, as shown in our linear regression.

A simple LSTM model with a small feature set seemed unable to capture the motion of tick direction accurately, though it removed the unstableness of testing accuracy, seen in the linear regression results.

Finally, we defined a LSTM model with a larger feature set. This model performed best, giving the highest testing accuracy and robustness for the results. Moreover, it converged very fast, and took only 1000 iteration. The effect of different parameters of the LSTM model was also analysed.