US Stock Market Prediction by LSTM

Author: Chris Tsai
E-mail: christsaizyt@gmail.com
Feel free to contact me if you have any comments or suggestions.

Introduction

In this repo, I would like to share some of my works using LSTM to predict stock prices. LSTM is long-short term memory network. I'm not going to the details on how LSTM works. For this, here is a fantastic article made by Colah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

LSTM is a very great choice to handle with time-series data rather than traditional Recurrent Neural Network (RNN). In RNN, there is a so-called gradient vanishing/exploding problem, and the problem comes from updating the weights by only multiplications. To solve the problem, LSTM considers another way to updating the weights not only by multiplications but also by additions.

In my work, I used two ways to do the predictions. One is stateless LSTM model and another one is stateful LSTM model. I'm still working on stateful LSTM model. I will update this part in the future.

Stateless LSTM model

Input

1. Get data from quandl (wiki database)
2. Features set = [*'Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close', 'Adj. Volume'*] (default feature set)
3. Create more features by default feature set (3-days MA, 5-days MA, or some technical indicators...)

Definition

Parameters
Using *window_len* days' historical features set to predict *pred_len* days later.

- *window_len*: append *windw_len* days' historical features set
- *pred_len*: predict the moving average(or close price) for *pred_len* days later
- *valid_len*: To do validation.

Data frame

- Divided time-series data into three parts: *df, df_valid, df_lately*
  a). *df -> X, y* (with label, for train and test)

b). *df_valid -> X_valid, y_valid* (with label, for evaluate the model, not for training)
c). *df_lately -> X_lately* (without label)

Output

1. Here we can use two types of output, regression and classification.
2. Regression: loss function is 'mse' for model
3. Classification: it can be easily done by applying KMeans algorithm to find the catogories. The loss function is 'categorical_crossentropy' for the model
4. There is another parameter in my model. 'out_type' could be 'MA' or 'close'. It means the target is moving average or just close price for how many days(*pred_len*) later.

Preprocessing

1. Divide the data for each row into price and volume.
2. Do standard normalization for price and volume separately.
   Here is an assumption: if winodw_len is large enough, it will be a Gaussian Distribution. Normalize to zero mean and unit variance.
3. Normalization for row data and do some data reshape
   Use sklearn preprocessing.StandardScaler()
4. Rearrange to original format
5. Save the *scaler_price* (it is necessary while doing the inverse transformation later)

Cross validation

In this part, time-series data will be shuffled. It means there is no stateful information for each row data.

- Split *(X, y)* into *(X_train, y_train) + (X_test, y_test)*
- *(X_train, y_train)* is for training the LSTM model.
- *(X_test, y_test)* is for test later.

LSTM model

[samples, time steps, features] = [how_many_data_u_have, window_len, n_feature_set]
Be care of the return_sequences

Regression

1. Build LSTM model with *input_dim = 5(ohlcv)*
2. Here I use two hidden layers *[120, 60]* with *dropout = 0.5*, *activation = 'relu'*
3. Output layer: *activation = 'linear'*
4. Loss = 'mse', *optimization = 'rmsprop'*

Classification

1. Build LSTM model with *input_dim = 5(ohlcv), output_dim = n_out_class*
2. Here I use two hidden layers *[120, 60]* with *dropout = 0.5*, *activation = 'relu'*
3. Output layer: *activation = 'softmax'*
4. Loss = 'categorical_crossentropy', *optimization = 'rmsprop'*

Validation and prediction

1. *df* for training / test

2. *df_valid* - I used it to evaluate the model. Sometimes the model can get a good training/validation loss but could not achieve the same performance for *df_valid*
3. *df_lately* - Predict the future trend.

How to use

1. There are two mean square errors in this work, *mse_test and mse_valid* based on *(X_test, y_test)* and *(X_valid, y_valid)*
2. If *mse_test* and *mse_valid* is close and small enough, (**maybe**) the model is good to predict for (*X_lately*).
3. Prediction results contain two columns: *'a_+10_d'*, *'p_+10_d'*

- *'a_+10_d'*: actual moving average or close price for 10 days later
- *'p_+10_d'*: predict moving average or close price for 10 days later

Stateful LSTM model

- Under development
- Please see the 'predictions' folder
- Train and predict a ticker will cost 8 hours (Linux mint 18.1, i5, 32G ram, GTX760 in tensorflow(GPU))
- The training process takes a long time, so I will keep updating more and more predictions results.