Summary of Stock Market Predictions with LSTM in Python Paper and Tutorial

Jody Shu

March 7, 2020

In this paper/tutorial, the authors gives details of how to code long-short-term-memory (LSTM) algorithm in Python. LSTM is one of the recurrent neural networks that evolve hidden states through time to capture the sequential pattern of input data. He started showing people how to download the time series data from Yahoo finance. After that, he split data into two groups, namely training and test sets. Finally the author showed how to implement a few averaging techniques and compared that with LSTM model which can predict more than one-step ahead. In this tutorial, he also has several helpful links for deep learning and averaging techniques.

There are different ways to acquire data from Yahoo finance, and one of the author suggested is to get a free api key and another way is to get it from his GitHub account which I tried both and was easier to get it directly from Alpha Vantage to get the key since I could not find the data in the GitHub account he mentioned. The example in the code is from American Airlines Group (AAL). First, the Python code sorted the data by date and output the table and graph the result. Then, he normalized the data using MinMaxScaler() method in Python to scale the data between 0 and 1, and then the data is splitted into train-test sets. Since I only total 3632 data, I take 80 % for training data (2906).

After that, the training set is divided into four windows and is applied the exponential moving average (EMA). The formula for EMA is as follows.
$S_t = \alpha \times Y_t + (1 - \alpha) \times S_{t-1}\ when\ t > 1$, and $S_t = Y_1$ when t=1
where α is a constant smoothing factor between 0 and 1, Y_t is the value at a time period t, and S_t is the value of the EMA at any time period t.
The other averaging technique in the tutorial is using normal/standard average. After these two algorithms are performed, the author compared their mean squared errors and plot the graphs verse the original/true data. It shows that EMA performs better than normal average. The author stated that this technique is good for the short term result (next few days) which is not so great to predict a longer future. Therefore, LSTM is far better than these averaging techniques which can predict the stock movement far into the future.
Then the author follows the below equations to write a class and feed the data in batch.

$$z_t = \tanh(W_z x^t + Q_z h^{t-1} + b_z)$$
$$i_t = \sigma(W_i x^t + Q_i h^{t-1} + b_i)$$
$$f_t = \sigma(W_f x^t + Q_f h^{t-1} + b_f)$$
$$c_t = f^t \odot c^{t-1} + i^t \odot z^t$$
$$o^t = \sigma(W_o x^t + W_h h^{t-1} + b_o)$$
$$h^t = o^t \odot tanh(c^t)$$

where $x^t \in \mathbb{R}^U$ is an input vector, $c^t\ and\ h^t \in \mathbb{R}^U$ denote the cell state vector and the hidden state vector, respectively, and U is the number of hidden units. Vector $z^t \in \mathbb{R}^U$ is an information transformation module. Vectors $i^t, o^t\ and\ f^t \in \mathbb{R}^U$ denote the input output and forget gate, respectively. $W_z, W_i, W_f, W_o \in \mathbb{R}$, and $Q_z, Q_i, Q_f \in \mathbb{R}^{U \times U}$ are mapping matrices; $b_z, b_i, b_f,\ and\ b_o \in \mathbb{R}^U$ are bias vectors.

Following the above mentioned equations, the author used three hidden layers and use the MultiRNNCell methods in TensorFlow to encapsulate the three LSTMCell objects, The author used for loop to run through three layers and compare the predictions and the true stock prices. And then the algorithm sum the mean squares error from each layer. Finally, he used Adam which is a newly developed well-perfoming optimizer. After that, the author used the test set to validate his algorithm which is pretty accurate.
All in all, this is a pretty informative paper/tutorial, however, since this is much more complicated than was taught in the class, it is very time consuming to tweak his code to make it work for me. However, there are partial code towards the end that I still need to figure out how to tweak it.