Machine Learning Engineer Nanodegree Capstone Proposal

Chao Zhou

25th December 2019

The stock market prediction has been identified as a significant practical problem in the economic field. Trading algorithms rather than humans performed over 80% of trading in the stock market and the FOREX market. In the crypto-currency market, algorithmic trading is also a hot topic among investors. However, timely and accurate prediction of the market is generally regarded as one of the most challenging problems, since the environment is profoundly affected by volatile political-economic factors, such as legislative acts, political unrest, and mass irrational panic.

There are many studies regarding algorithmic trading in financial markets based on machine learning, where recurrent neural network (RNN) and reinforcement learning (RL) are being popular in recent years. In this study, a Bitcoin price predictor based on long short-term memory (LSTM, a variant of RNN) is presented.

1 Problem Statement

Given a time-series data of Bitcoin price with each time step indicating one minute, the goal is to build a price predictor of the price of the next minute.

2 Datasets and Inputs

In this study, data of the BitMEX's XBTZ19 contract, which is a Bitcoin futures contract expiring in December 2019, is used to train and test the

predictor. The dataset is a table that each row indicates one minute and each column indicates a specific data such as open-high-low-close prices, volume, and volume-weighted average price (VWAP). Data could be fetched from BitMEX's official API without charge of fee. The API concerned with the desired data is documented at `https://www.bitmex.com/api/explorer/#!/Trade/Trade_getBucketed`.

The dataset is formulated as $\{x_t | t = 1, 2, \ldots, T\}$, where x_t is a vector of the data in minute t, such that

$$x_t := \begin{bmatrix} \text{open price of time } t \\ \text{high price of time } t \\ \text{low price of time } t \\ \text{close price of time } t \\ \text{volume of time } t \\ \text{VWAP of time } t \\ \vdots \end{bmatrix}.$$

3 Solution Statement

Given data of the time range with length of N and end at time step t, viz $\{x_{i-N+1}, \ldots, x_i\}$, the predictor estimates the VWAP of the next time step $y_i = v_{i+1}$, where v_{i+1} is an item of vector x_{i+1}. The predictor can be seen as a function $\hat{y}_i = f(x_{i-N+1} \ldots x_i)$ mapping from the history trading data to the next VWAP.

The predictor will be an LSTM model implemented with PyTorch.

4 Benchmark Model

A linear regression model will be used as the benchmark to compare against the LSTM model.

5 Evaluation Metrics

The mean squared error (MSE) between labels y and predictions $\hat{y}$ will be used to evaluate the performance of both of the benchmark model and the solution model. For a given N and a time-series test dataset, all consecutive sub-sequence of the time-series with length N will be used and equally contribute to the final MSE.

6 Project Design

This project consists of the the following parts:

1. a crawler that fetch data from BitMEX's API;
2. a feature engineering script;
3. an LSTM model implemented with PyTorch;
4. a training script executing backpropagation; and
5. a evaluating script.

6.1 Crawler

Data returned by BitMEX's API in JSON format will be parsed as a table and saved to disk for further manipulation.

6.2 Feature Engineering

The API returns 12 fields for each time step (or minute in this case). Only 5 of them are needed as features for the machine learning model, which are high price, low price, close price, volume, and VWAP. The open price returned by BitMEX API is defined as the close price of the last time step instead of the price of the first trade during the specific time, and therefore will not be taken as a feature. Other fields, such as instrument symbol, are irrelevant and will be abandoned.

Technical indicators, like moving average convergence/divergence (MACD), relative strength index (RSI), will be used as additional features.

Consisting of five original columns and technical indicator columns as features, the entire time-series will be horizontally split into three consecutive parts for training, validation, and testing, with lengths proportionate to 6:2:2.

6.3 Models

The training dataset will be sent to train the benchmark model and the solution model. Both models are implemented with PyTorch and trained at AWS SageMaker.

The benchmark model consists of two linear regression layers:

$$x_1 = \text{ReLU}(xA_1 + b_1)$$
$$y = \sigma(x_1 A_2 + b_2)$$

The solution model consists of one LSTM layer and one linear layer. The LSTM layer is formally formulated as follows:

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned}$$

where where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function, and $*$ is the Hadamard product.

6.4 Evaluation

The real VWAP, predicted VWAP, and the MSE among the entire time range will be plotted as diagrams to demonstrate the achievement of this study.