Undergraduate Research Fellowship (SURF)

Research Journal

Woo June Cha

Carnegie Mellon University

https://github.com/hyunjune123/Q-trader.git

Title : Implementation of Q learning in stock markets

Note : This research is still in progress.

Abstract

It is impossible for an empirical model or algorithm to completely harness the unpredictive nature of the market. Nonetheless, it is not completely impossible to gauge the general trend and predict changes in the market by technical analysis. Specifically, this research will explore the potential of Q learning applied in stock markets, by training a Q learning based model so that it can emulate the "Bollinger Band Squeeze Strategy".[1] Historical stock data will be gathered and meaningful input features for the model will be extracted. The parameters of the model will be tuned as the model is trained with input features from stock data. The model will be tested on the most recent stock data and its performance will be recorded. Hopefully, the results will demonstrate the potential use of Q learning in market decisions.

Research Question

The Federal Reserve reports that more than half of non-retirees do not participate in any kind of investments with their retirement accounts. The level of comfortableness in investing rises with education level and financial knowledge. For instance, people with Bachelor degree or above employ wide ranges of financial tools with confidence, while the others find it hard to even access the same tools.[2] Investment Decision is somewhat closely related to financial literacy, and thus people with limited financial resource tend to restrain themselves from investing.[3]

There is an ongoing effort to contrive decisive algorithmic trading models based on technical analysis. These algorithms will aid the investment decisions of those who lack financial backgrounds. One example model is a Q learning based model, which is a model that learns how to make certain trade decisions in appropriate situations by running trade simulations on historical data. The application of Q learning in the fields of finance is in its early development. However, with the technological advancement in deep learning, research on incorporating the proper features and architectures for the Q learning model is becoming more prevalent.[4]

The specific goal of this research is to come up with a profitable and stable Q learning trade model that could successfully learn a popular trading strategy called the "Bollinger Band Squeeze Strategy". Ultimately, if Q learning models become sophisticated enough to guarantee high and stable profit with enough research

---

[1] "Bollinger Bands." Bollinger Bands [ChartSchool], school.stockcharts.com/doku.php?id=technical_indicators%3Abollinger_bands.

[2] "Report on the Economic Well-Being of U.S. Households in 2018 - May 2019." Board of Governors of the Federal Reserve System, May 2019, www.federalreserve.gov/publications/2019-economic-well-being-of-us-households-in-2018-retirement.htm.

[3] Ronald Brownstein, National Journal. "Why Some Americans Buy Stocks and Others Don't." The Atlantic, Atlantic Media Company, 21 Nov. 2013, www.theatlantic.com/business/archive/2013/11/why-some-americans-buy-stocks-and-others-dont/426243/.

[4] Dai, Yuqin, et al. Reinforcement Learning for FX Trading. Stanford University.

and back testing, its applications would certainly encourage more people to start investing on their future. Even if the model does not turn out to be as profitable, identifying the flaws of the model would be meaningful for future research on Q learning applications.

Project Design and feasibility

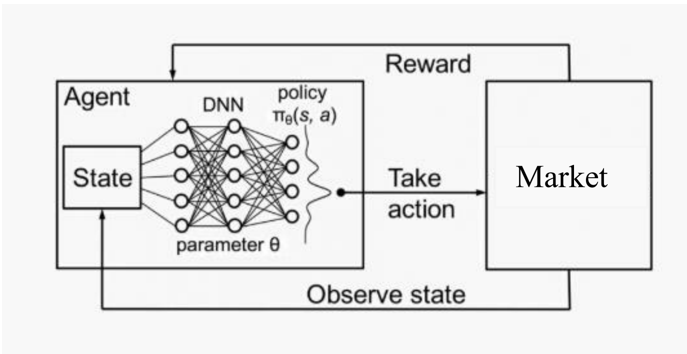
Strategy

The trading strategy used for the model is based on the Bollinger Band Squeeze Strategy. This popular strategy is based on the belief that price variation occurs when a "Squeeze" phase ends. The "Squeeze" phase is the time period when the Bollinger band is within the Keltner band. In the image below, the yellow shaded region is the "Squeeze" phase. After the squeeze is over, traders sell when price fall below the Bollinger band, and buys if price goes above the band.

Source : https://admiralmarkets.com/analytics/traders-blog/bollinger-bands-r-with-admiral-keltner-breakout-strategy

If a statistical model learns how to successfully recognize a "Squeeze" phase, the model can emulate an optimal trading decision. This is where Q learning kicks in.

Q learning



We will train the model so it can learn how to make an optimal decision given some states or characteristics of the market. The imaginary agent (or trader) will observe the market to get useful states of the market. The expected future profits of taking each action (buy, sell, or hold) will be computed by a neural network (DNN), to determine an action. Once the agent takes the action and gets rewarded (profit or loss) from that action, the neural network will be updated so that next time it computes more accurate expected future profits. In other words, the model remembers the market state, its action, and the result of an investment, and uses that memory next time. Each components of the model of interest would be the following:

- State : Moving averages, Bollinger band on price and volume, Keltner Band, RSI, etc
- Action : {buy, sit, sell} (or could be discretized to buy and sell at 0% ~ 100%)
- Environment : Stock's historical data
- Reward : Sell : (sold $ - bought $), Buy : some constant x, Hold : 0

- Q value approximator : Default DNN, but LSTM or RNN could be an option

Checkpoint (7/27)

Initial model

Technical analysis on stock prices require indicators of 3 categories: (1) moving trend, (2) volume, (3) magnitude of price changes. For each categories, using one indicator is enough, otherwise the inputs would get redundant. We use (1) 20 days moving averages + bolinger bands, (2) Volume, and (3) RSI for the Q-learning model.

(1) 20 days moving average + bolinger bands (b)

- Gives info regarding the price's position in respect to the moving average and bolinger bands.
- Input = (Price - Moving average) / (Standard deviation of 20 typical price)
- Typical price = (Low + High + Price) / 3

(2) Volume (v)

- Percent increase from previous volume
- (V(t) – V(t-1)) / V(t)

(3) RSI (r)

- https://www.investopedia.com/terms/r/rsi.asp

The final input line would be an array of length 30 with normalized 10 entries of each categories …

```python
def make_train_line(df):
    bol = (df["Bolinger"].tail(10)).to_numpy()
    bol = bol / np.linalg.norm(bol)
    rsi = (df["Rsi"].tail(10)).to_numpy()
    rsi = rsi / np.linalg.norm(rsi)
    vol = (df["Volume"].tail(10)).to_numpy()
    vol = vol / np.linalg.norm(vol)
    state = np.concatenate((bol, rsi, vol))
    return state
```

State = [b1, b2 … b10, v1, v2, … v10, r1, r2, … r10]

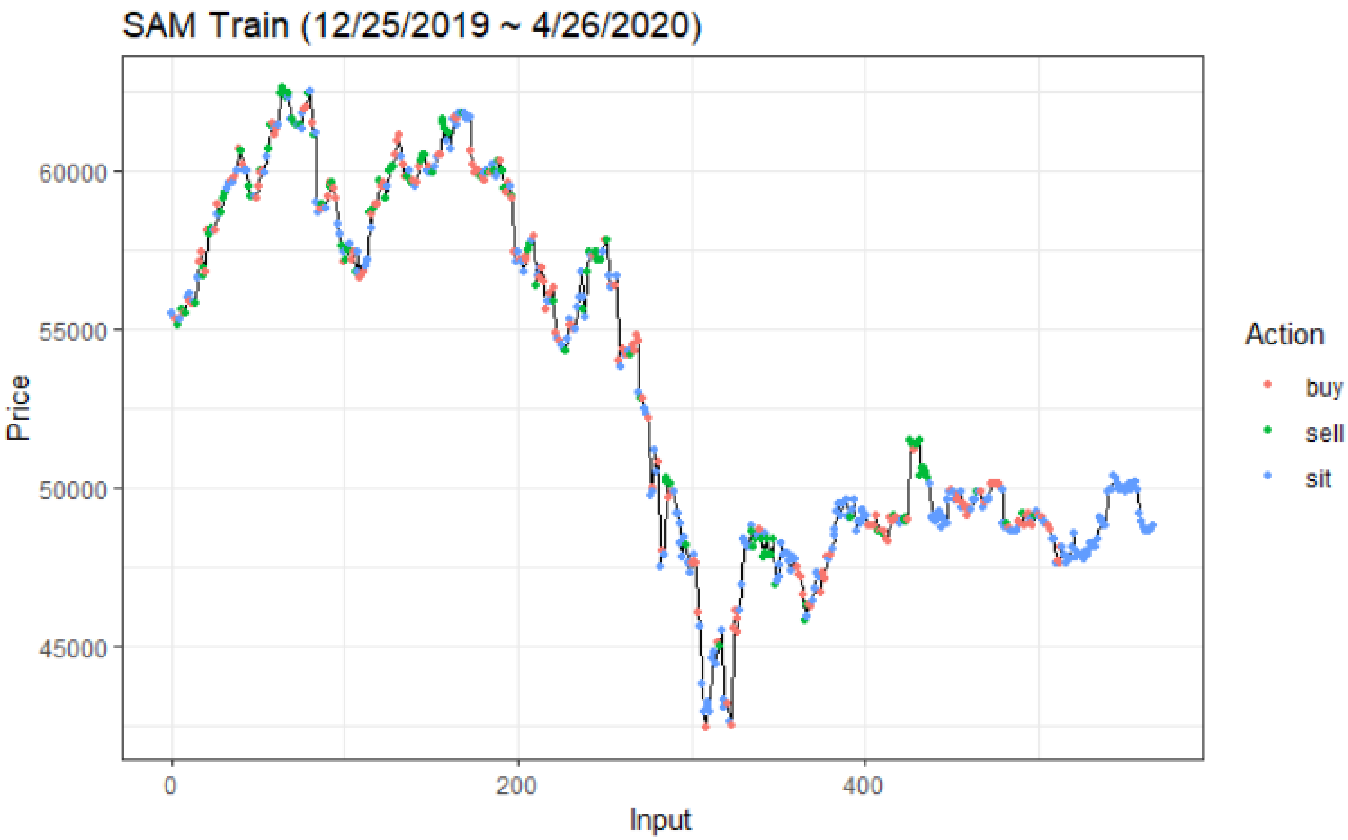
When the model decides to buy, it will buy stock worth half of its current cash. When it decides to sell, it will sell only half of its position.

Transaction cost/commission was set to 0.75 % of transaction.

Checkpoint (8/8)

Training

Each time the model make decisions based on its memory (trained neural network) and past 10 days' indicators. We are hoping that the model can correctly catch buy or sell signals after the squeeze phase mentioned above in the proposal. The figure below is the result of a training phase on SAMSUNG's hourly stock price from 12/25/2019 ~ 4/26/2020. For convenience, dates were replaced with index on horizontal axis.



The first 100 or so inputs should be disregarded since the model was learning with highly randomized decisions for exploration purpose. When prices hit peaks, we see that the model tends to sell (green) more often. Pattern for buys (red) seems to be below the sells (green), which is what we wanted. The model does grasp the price changing trend.