A Trading Algorithm Based on Deep Q-Learning

Shen Gao (shengao)

Abstract

Quantitative investing has become the new norm in today's financial markets. According to a recent study, about 70% of overall trading volume is generated through algorithms. It has become crucial for not only the savvy institutional investors to leverage complex trading system to compete in the sub-millisecond latency space but also retail investors who could invest effectively in this new dynamic.

In this paper, we examine a deep-learning based trading algorithms and discuss its ability to make investment decisions.

Dataset

Currently I'm relying on daily price time series from Yahoo Finance and alpha vantage API for stock prices. The alpha vantage API also provides intraday minute level tick data on not only major stocks but also cryptocurrencies and foreign exchange. In the next phase of the project, I will train the model on minute level time series data as well as across multiple other asset classes.

Approach

Under the Q-learning framework, we define the following basic components:

- State (s): number of stocks owned, current stock price and total wealth
- Action (a): buy 1 stock, sell 1 stock or hold/do nothing
- Reward (r): profit or less by taking the action
- Policy (Q): what the Q network is trying to figure out

The Q network updates as follows:

$$Q(s, a) \sim r + \text{gamma} \times Q'(s', a')$$

where gamma is the discount factor, s' is the next optimal state, a' is the optimal action in the next state, r is the immediate reward associated taking action a'

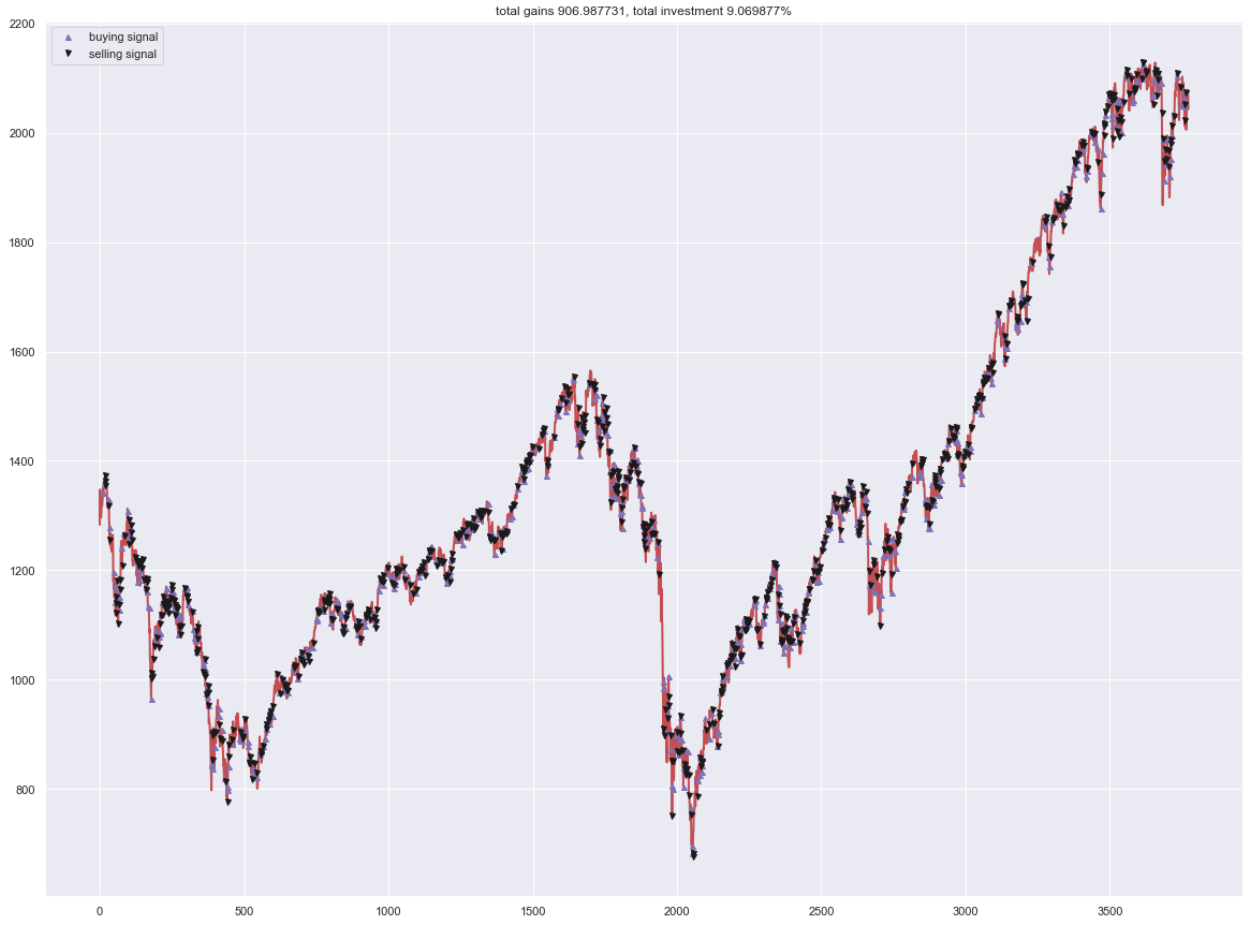
Psedo code for the Q-learning algorithm:

Initialize Q(s, a) randomly
Repeat (for each epoch):
    Initialize s
    Repeat (for each state):

Choose a that the agent employs to determine next action ($\text{argmax}_a Q(s, a)$)
Take action a, observe r, s'
Update
Q(s, a) = Q(s, a) + r + gamma x max(Q(s', a') – Q(s, a))
s = s'
Until state reaches the end
Until iteration exhausts or model converges

Preliminary Results

Buy-sell actions:



The model returns a small positive return, but as depicted the trading decisions are made very frequent. I would be interesting to include consideration for trading cost as that would significantly reduce the rate of action.

Future Work

The next phase of the project will be focused on the following aspects:

1) Use minute-level tick data to dramatically increase the data set

2) Consider altering actions to include rebalancing to a different stock or a different asset class. This will dramatically complicate the step where optimal action is determined.
3) Introducing other state variables to augment the agent's ability to observe from state variables
4) Restructure code to run on AWS
5) Consider a different reinforcement learning approach for comparison

Code

https://github.com/ggaoshen/Stanford_CS230_DL/tree/master/Project

Reference

Paper: http://cslt.riit.tsinghua.edu.cn/mediawiki/images/5/5f/Dtq.pdf
GitHub repo: https://github.com/kh-kim/stock_market_reinforcement_learning