Deep Learning Project Progress Report

Hüseyin Talha Şenyaşa[1, *]

[1] *Department of Physics, Faculty of Science and Letters, Istanbul Technical University, 34469 Maslak, Istanbul, Turkey*

My personal notes on *An application of deep reinforcement learning to algorithmic trading.*

I. IMPLEMENTATION OF THE FRAMEWORK

Some definitions and symbols

1. S: Set of environment and agent state.
2. A: Set of actions which are available for agent to use.
3. s_t: RL environment internal state
4. o_t: Observation
5. a_t: Trading action
6. i_t: Information
7. $\pi(a_t|i_t)$: Trading policy (Rule)
8. r_t: Network's reward.
9. ν_t^c: Total amount of cash in portfolio.
10. ν_t^s: Corresponding value of the share.
11. n_t: Total number of shares, lots.

Reinforcement learning techniques are concerned with the design of π maximizing an optimality criterion, which directly depends on the immediate rewards r_t observed over a certain time horizon.

A. Trading Environment

The nature of an environment in deep reinforcement learning applications is dictated by the problem in question. In our case, the medium of the environment consists of discrete time series indexed by time-step t. The interval, between sequential data points (corresponding to $t-1$ and t) is determined by the trading frequency and it is denoted by Δt. Since we only consider daily trading, the interval Δt is equal to 1 day.

The information that is available to the agent for observation consists of two parts. The first part is the classical OHLCV (*Open, High, Low, Close, Volume*) data of the stock in question, and the second part is the trading position which is denoted by P. Formally the observation made by the agent at time-step t can be expressed as

$$\mathcal{O}(t) = \{p_t^O, p_t^H, p_t^L, p_t^C, V_t, P_t\}, \tag{1}$$

where

- p_t^O is the opening price of the stock over the period $[t-\Delta t, t]$,
- p_t^H is the highest price over the period $[t-\Delta t, t]$,
- p_t^H is the lowest price over the period $[t-\Delta t, t]$,
- p_t^C is the closing price over the period $[t-\Delta t, t]$,
- V_t is the total volume of shares exchange over the period $[t-\Delta t, t]$.

Initialization, iteration and reset. In the initialization part, one first builds the environment in a state which can be determined randomly. This initialized state corresponds to current status of the environment. In the iteration part, first, the RL agent provides an *action* to take, then, the environment state is transitioned to the next state according to the action provided by the RL agent.

1. Description of Trading Operations / Action Space

The agent starts its trading activity with an initial C_0 amount of money and it is chosen in such a way that it is much lower than the average exchange volume of the stock in question. In this way, it becomes safe to assume that the actions carried out by the agent does not influence the stock movements.

For a given time-step t, the agent can take two actions: *long* and *short*. A long action results in purchasing shares of the stock while a short action results in selling shares of the stock in question.

Let C_t be the available cash at time-step t and let Q_t be the number of shares the agent is going to buy or sell. If the agent takes the long position at time-step t, the agent will have

$$Q_t = \left\lfloor \frac{C_t}{p_t^C(1+T)} \right\rfloor, \tag{2}$$

2. Implementation of Trading Operations

One first needs to implement the trading operations to be able to implement the trading environment. The trading operations correspond to the actions that is taken by the agent. In normal circumstances, there must be a match between bid and ask orders to realize a trade operation. In order to detect these matches, one must be able access the order book of the stock in question.

* senyasa@itu.edu.tr

However, since assume that the total number of buy and sell orders are much smaller than the volume of the stock, we are going to implement only buy and sell operations.

The trading operations are implemented in an object oriented manner. There are two main classes to control trading operations: *StockHandler* and *DummyPosition*. The StockHandler class obtains the stock data and applies minor transformations to eliminate possible missing data points. This class also responsible for the real-time tracking of the stock via *Update* method. The DummyPosition is class is implemented to carry out possible actions i.e., *long* or *short* operations. and to keep track of the position changes. Here we provide the prototypes of the two classes.

StockHandler:

3. Implementation of Trading Environment via OpenAI Gym

The environment which the deep learning agent interacts with is implemented by utilizing *OpenAI Gym* framework. OpenAI Gym is an open source python library that provides standardized application programming interface (API) to establish interaction/communication between agents and environments for reinforcement learning problems.

A third party application has to implement/override several functions/methods that are inherited from the *gym* base class. These methods include initialization, iteration and reset of the environment in question.

B. Implementation of Deep Neural Network

Network Architecture, loss function etc.

C. Implementation of Double-Q Mechanism

II. ADDITIONAL NOTES AND QUESTIONS

1. Representing the transition from one candle to the next one as a Markov process.
2. Considering the correlation between candles as a spin system (as in the case of Witten's "An introduction to quantum information theory")