Environment:

This is a single stock environment. It means that at each episode one single stock is **randomly** selected from a pull of stock data. I have selected 25 stocks from Indian market which have high volume and also high beta.

Past 3 months historical data is used for training. And next (from test data time) 1 week data is used for testing.

1. Observation:

The observation space have the following 12 parameters:
1. Day, 2. Hours(Time), 3. Minutes(Time) 4. Open prices, 5. High prices 6. Low prices, 7. Close prices, 8. Volume, 9. Money reserve, 10. Stock holding, 11. Action taken at previous step, 12. Reward at previous step

The observation will look like as follows:

[[Day_t-n, Hours_t-n, Minutes_t-n, Open_t-n, High_t-n-1, Low_t-n-1, Close_t-n-1, Volume_t-n-1
, Money Reserve_t-n-1, Stock Holding_t-n-1, Action_t-n-1, Reward_t-n-1],

[.....],

[.....],

[Day_t, Hours_t, Minutes_t, Open_t, High_t-1, Low_t-1, Close_t-1, Volume_t-1, Money Reserve
_t-1, Stock Holding_t-1, Action_t-1, Reward_t-1]]

It is a [12*n] 2d numpy array where n is the look_back_window.

So, the observation space should look like as follows:
spaces.Box(low=-np.inf, high=np.inf,
shape=(self.look_back_window, 12),
dtype=np.float64)

****
Can I neglect the close price ? or open price ?
should I put month in observation?
I will incluse macroeconomics data such as Nifty and Sensex points, Dollar Exchange rate.
****

2. Action:

This is a long-only environment. No shorting action is allowed.

Action space is discrete +ve integers. It is scalled in following way internally e.g., [0 to 11] is scalled [-5 to +5].

Position_per_action_value = percentage of fund will be used to buy or sell per unit int value in action.

Example: For an action space [10 to -10]. If action value is 2, 20% of the cash balance will be used to buy stock. Also, if action value is -4, 40% of the stock holding will be sold.

3. The trade execution mechanism (Indian Market):

3.1. Random selection of a stock: At every episode, a random stock is selected from the list of stocks, i.e., list_of_stock.

3.2. Stock data: The environment needs historic market data (.csv files) in OHLC format which is saved in the data directory, i.e., data_dir.

3.2.1. The index column must be in datetime format and named as 'date'.

3.2.2. The input dataframe must have 'open', 'high', 'low', 'close', and 'volume' columns

3.3. Random time duration: At every episode, a random time duration of the episode length from the data frame is selected. The length of the episode (episode_length_in_steps) is user-defined.

3.4. Trading Fee: At every step, a trading fee is needed to pay. The total fee is ((trade_volume_in_money × fee_percentage)/100) or max_fee, whichever is the minimum.

3.5. Execution Price: The execution price is a random price between the high and low price of the time step.

3.6. Buy and Sell mechanism: In India, fractional stock buy and sell is not allowed.

3.6.1. Buy action execution: Buy action is executed when action_value > 0

The target position is given by,

target_position_in_money = self.money_reserve × (1 – (self.max_fee /100)) × (action_value/100)

The term, (1 – (self.max_fee /100)) ensures that there is enough money left to pay the fee. When the fee is much smaller this approximation should not affect the environment significantly.

The actual stock buy quantity is

stock_buy_in_quantity = int(target_position_in_money / execution_price)

3.6.2. Sell action execution: Sell action is executed when action_value < 0

The stock sell quantity is given by

stock_sell_in_quantity = int(self.stock_holdings_in_num × (-self.action/100))

3.7. Episode Ending: An episode ends is the maximum number of steps, i.e., (episode_length_in_steps) is over of the net_worth is reduced more than the acceptable loss, (max_acceptable_drop_down_percentage_episode_stopping).

4. Reward:

4. Reward:

Reward is function is designed as follows

Reward(t) = ln(net_worth(t)/self.old_net_worth(t-1))

I have adapted it from Gym Trading Environment [1]

[1] https://gym-trading-env.readthedocs.io/en/latest/environment_desc.html#reward

Seems to be legit because

$$\ln\left(\frac{a}{b}\right) = \ln(a) + \ln\left(\frac{1}{b}\right)$$

So, if all the reward of all the time steps of an episode are summed it will be

$$\ln\left(\frac{\text{Net\_Worth\_t}_1}{\text{Initial Balance}}\right) + \ln\left(\frac{\text{Net\_Worth\_t}_2}{\text{Net\_Worth\_t}_1}\right) + \cdots + \ln\left(\frac{\text{Net\_Worth\_t}_{end}}{\text{Net\_Worth\_t}_{end-1}}\right)$$

The summation will be equal to

$$\ln\left(\frac{\text{Net\_Worth\_t}_{end}}{\text{Initial Balance}}\right)$$

However, I have to explore more for finding better Reward Function.

Seems to be legit. However, I have to explore more.

Cause ln(a/b) = ln(a) + ln(1/b)