Batu Helvacıoğlu
Can Gözpınar
Koray Tecimer

BaCaKo Trading Algorithm Final Report

Problem Definition:

Aim of this project is to apply reinforcement learning to train agents to trade lucratively in the stock market. Our implementation comprises two agents that are trained using deep reinforcement learning policy gradient that are specialized in terms of the time intervals that they are focused on. One of the agents would be trained to maximize its reward function with respect to intervals of 24 hours in other words, in a short-sighted manner compared to the other agent. The other agent would be trained on weekly data which would result in the agent to favor actions which would maximize its reward in the long run. We assume that the short term and long-term models will sometimes differ in their predicted actions. We want to explore and use these differences to create a better model. These two models will be used to create intrinsic rewards for further iterations of the models. New short term and long-term models will be trained with intrinsic rewards. Intrinsic rewards will be calculated from the comparison of the training model's actions versus the previously trained models. For example, the long-term model with intrinsic reward will use the normal short term model's predictions to create the intrinsic rewards. If the predictions/actions of the models do not match, a positive reward will be given as an incentive to explore this state further. By creating this separation between the agents and the interaction between them we hope to reduce overfitting and explore critical areas of the market that might have potential for profit. Finally, these core trained agents would output actions that correspond to selling, holding, or buying stocks and the amount to do so.

Problem Formulation:

Our problem will be episodic, we will use 28 workday and 33 workday episodes for short- and long-term models, respectively. All buy and sell operations will be executed at market price.

We will have two distinct state representations for two agents both containing data about a stock. The long-term agent will have a broader understanding about the stocks whereas, the short-term agent will focus on a shorter duration of time.

The short-term agent will have the candlestick data of the last trading day for every 5 minutes. This means the short-term agent will know the opening, closing, highest and lowest price of the stock for every 5-minute interval in the last trading day when the exchange is open.

The long-term agent will have the data for the last one week. It will have the opening, closing, highest and lowest price of the stock for every hour in the last five workdays.

All stock prices will be normalized to the most recent data point's opening price (current price). Input state of a given date and time consists of 105(short term) or 40(long term) previous time steps of stock price and trading value data. All of the previous prices and trading volumes are normalized with the opening price of the given date and time. For example, 104^th row could be 0.99 0.97 0.99 0.96 0.99 while 105^th row is 1 0.98 0.99 0.97 1. The 105^th row is the data of the given date and time therefore, the opening price and the trade volume of 105^th row is 1 since it was normalized with these values. This normalization allows us to focus on the relation between the time steps instead of the actual price values. Both agents will know the amounts of assets in proportion to the initial assets.

Actions will be selling the stock, holding the stock and buying the stock. These actions will be continuous because along with sell, hold and buy commands, the percentage of the stock to sell, or the percentage of cash to buy with will be determined by the agents. The 3 actions will come from a softmax activation function. If the hold action is the maximum, the agent will not do anything. If either buy or sell is higher, the higher action will be taken with the percentage of the action's value. For example, if the sell action has a value of 0.70, 70% of the stock will be sold and 30% of it will remain. If the buy action has a value of 0.70, 70% of the cash budget will be used to buy the stock.

This will ensure that the agent will remain conservative with its actions by preferring the hold action. Intrinsic rewards will not be affected by these action definitions as they will use the numerical outputs from the networks. Continuous nature of the actions is required to be able to accurately compare the predictions of the short- and long-term models.

External rewards will be the difference of total returns normalized to the initial total assets between each state. For example, if the agent starts with 1000$, has 2000$ in state A and then transitions to state B where it gains 200$ the reward will be 2200/1000 - 2000/1000 = 0.2. Along with the external rewards our agents will receive intrinsic rewards in the later iterations to encourage exploration. Intrinsic reward will decrease as the similarity between the actions of the long term and the short-term agent increases. This would allow the agents to explore the actions that they would not normally take. Important thing to be aware of at this point is that the intrinsic rewards should not outweigh the external rewards because we do not want agents to make wrong decisions just to take a different action from the other agent. Difference between the decisions of the model will be calculated by taking the difference between their predictions, taking the square of these differences for each element of the output vector. Weighted sum of these squared differences will give the intrinsic reward. The difference between the decided action will have the highest weight while the difference between the predictions for each action type will have less weight.

$$A_s = action\ vector\ for\ short\ term$$
$$A_L = action\ vector\ for\ long\ term$$
$$W_0 = weights\ of\ squared\ differences$$
$$W_1 = weights\ of\ squared\ differences$$
$$\varepsilon = intrinsic\ reward$$

$$\varepsilon = (A_s(0) - A_L(0))^2 * W_0 + \sum_{i \in \{1,2,3\}} (A_s(i) - A_L(i))^2 * W_1 \qquad W_0 > W_1$$

$$c_i = initial\ capital$$
$$s' = next\ state$$
$$I_s = total\ capital\ in\ state\ s$$
$$R_s = reward\ observed\ in\ state\ s$$
$$\varepsilon = intrinsic\ reward$$

$$R_{s'} = (I_{s'} - I_s)/c_i + \varepsilon$$

We will use A2C for their advantages such as faster convergence, better exploration, and stochastic policy.

Related Work

The intrinsic reward to encourage exploration was suggested by Deepak Pathak, Pulkit Agrawal, Alexei A. Efros and Trevor Darrell in the paper (2) Curiosity-driven Exploration by Self-supervised Prediction. In this paper an unsupervised model alongside the reinforcement learning model was implemented to create the intrinsic rewards. The unsupervised learning made a prediction of the upcoming environment after the action is taken. An intrinsic reward is calculated according to the contrast between the prediction and the actual environment. This positive reward encourages the model to explore states/environments that are new. We based our project on this idea of encouraging exploration of unexpected states. However, actions of our agent in the stock market will be negligible compared to the state of the market. We cannot change the market with our actions. Therefore, we decided to use two reinforcement learning models, one trained with short term data and the other with long term data. These two models will be used in the intrinsic reward module. Predictions/actions from these two models will be compared with the training model's actions to create intrinsic rewards.

For example, while training the short-term curiosity driven model, the long-term model's predictions will be used to create the intrinsic rewards. And while training the long-term curiosity driven model, the short-term model's predictions will be used. If the actions of the curiosity driven model does not match up with the opposite standard model (no intrinsic reward mechanism), a positive intrinsic reward will be given to encourage further investigation of these states by the model.

We referred to the paper (1) Reinforcement Learning for FX Trading by Yuqin Dai, Chris Wang, Iris Wang, Yilun Xu of Stanford University while deciding on the action space and deciding on the type of reinforcement learning algorithm to use. In this paper they made a comparison between DQN's and policy gradients. We decided to use policy gradients and a continuous action state because of their comparison. A continuous action space further complements our idea of comparing actions from two models.

Implementation Details:

We will use python as our programming language. As for the libraries we will use keras for generating neural networks, tensorflow, tensorflow probabilty, matplotlib, numpy and pandas. We created the environment ourselves. We used an off-policy A2C algorithm. We used batch learning to prevent our Neural Network from diverging instead of converging to the desired behavior. For training both of our agents we used data from 1 January 2020 until 2021 June of Gazprom's stock in Moscow Stock Exchange. For the short-term agent, the interval between each data point is 5 minutes whereas for the long-term agent we will use 1-hour intervals. Each episode both short- and long-term agents start the episode with random proportions for value of assets and investable capital and a random start date for episode between 1 January 2020 and 4 May 2021. We used the data from 4 May 2021 to 1 June 2021 as validation. Each episode we store the transitions and at the end of the episode the agent improves based on transitions for that episode. We randomized the start state to train the data with different time frames. Each state will have the last 24 hours' opening, low, high, closing data points for every 5-minute interval for the short-term agent and last 1 week's opening, low, high, closing data points for each hour for the long-term agent. State's prior opening, low, high, closing data points will be normalized according to the most recent opening price of the Gazprom stock. This is because neural networks work better with scaled data and the relation between the data points are more important for our agent than the actual price values. Each episode lasts 28 days for the short-term agent and 33 days for the long-term agent. We train our agents 1000 episodes each. Our network has a discount factor of 0.99. Our agents will use epsilon greedy action selection with epsilon as 0.2 and learning rate as 10^-6. We used a low learning rate because higher learning rates caused exploding gradients. Our neural networks have 3 dense layers, one input layer and one output layer in both the short term and the long-term versions. Short term model has an input dimension of 527, 105 for 5 min intervals times 5 data corresponding to opening, high, low, closing prices and trade volume plus 2 for cash and stock amounts. Long term model has an input dimension of 202, 40x5 + 2. First 2 dense layers have dimensions 512 and 256, respectively. Both use relu as activation function. The third dense layer has a dimension of 1 and uses linear activation. This layer is used for the value function approximation output. The other output layer is for the action probabilities and uses softmax. Value function approximation output is the critic part of our agent, and the action output is the actor part. Our agent further processes the action outputs to convert them to the form we mentioned in our problem formulation. We use epsilon greedy for exploration. Our agent returns an action vector of dimensions [1,4]. First element in this vector is the chosen action's probability followed by the probabilities of all 3 actions in the order of buy, hold, sell. Environment uses these probabilities in the intrinsic reward calculation and for the buy/sell amounts. Intrinsic reward weights were determined by trial and error. The score at the end of the episode represents the proportion of the initial budget gained. We tried to choose the weights of the intrinsic reward so that the score at the end of the

episode will be less than 1. Weight of the chosen action meaning the first element in the action vector is 1e-4 while the other elements have weight 1e-5.

Evaluation Plan:

The performance of the algorithm would be evaluated solely on the rate of return. We would take the long-term agent's and the short-term agent's rate of returns as the benchmark. The benchmarks are the ones that are trained without receiving the intrinsic reward. Benchmark values would be compared against the corresponding specialized agents that received intrinsic rewards during their training. By doing the evaluation process this way, we hope to establish a clear understanding of the impact that the intrinsic reward has on the model's behavior.

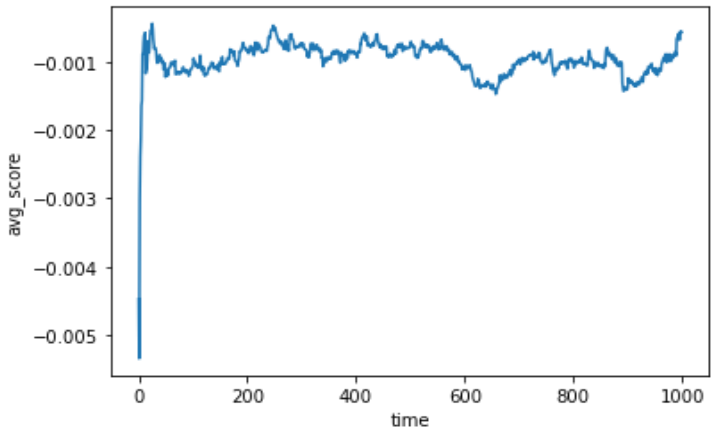
Validation:

We reserved a portion of our Gazprom stock data for validating our agent. We test our trained agent in a new time frame to see how well it performs under new conditions. We used the same starting date of 4 May 2021 for validation but initialized the budget distribution between cash and stock randomly. We ran the validation 100 times for each model.

Results & Discussion:

Our long-term agent without the intrinsic reward performed poorly according to our evaluation plan. For 1000 episodes the agent's average score was never above 0 and this indicates that our agent was unable to make profit on average. Although our agent failed to make a profit, it was able to improve itself. Our agent started with 0.5 %loss on average and increased it to upwards of 0.1% loss. It is also important to note that the results we observed were highly affected by the state that the agent started from which was randomized. We got higher and lower returns with different runs. Figure 1 is the rolling average score of the last 100 episodes of the long-term model without intrinsic rewards during training.

Figure 1.



Average score of long-term agent with no intrinsic rewards present

Figure 2 is the average scores of long term no intrinsic reward model in the validation set. It averages a loss of 2% which is significantly higher than the training results.

Batu Helvacıoğlu
Can Gözpınar
Koray Tecimer

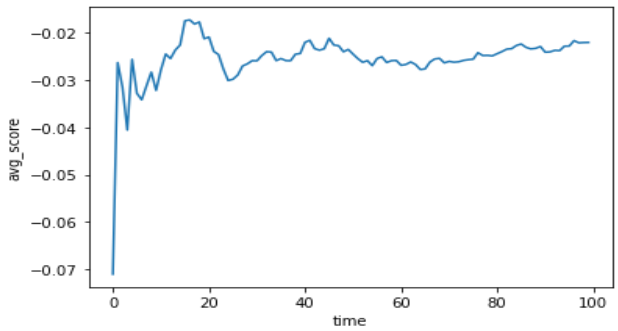


Figure 2.
Long-term agent no intrinsic reward average validation per episode

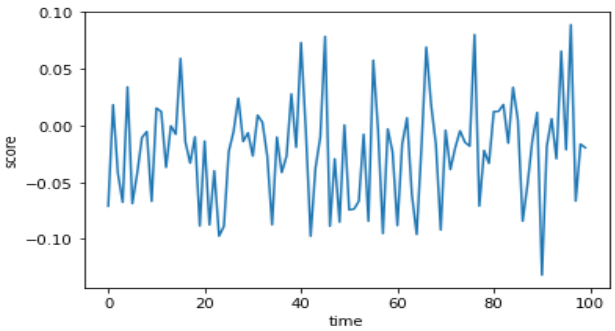


Figure 3.
Long-term agent score per episode

Figure 3 is the scores of each run of the long term no intrinsic reward model in the validation set. The random initialization of the budget allocation greatly affects the model's performance. This might be due to the model focusing on a single action, and that action being more profitable for a certain budget allocation between cash and stock while not being appropriate for an opposite budget allocation.

Long term model with intrinsic reward, short term model with no intrinsic reward and short term with intrinsic reward are still being trained. We will update our submission as soon as the training is over.

Conclusion:

To further improve our algorithms, we can use multi-step updates with prioritized replay to select the states which have the potential to make bigger improvements on our policy to learn from. Instead of prioritized learning one can choose to use a replay buffer. Another improvement can be fine tuning the weights used in the intrinsic reward calculation. We could not do a very good job on tuning these weights since time did not permit us to experiment much with changing their values due to long training hours required. Our intrinsic rewards may be overpowering the external rewards. We can also decrease the intrinsic rewards as the model trains since its aim is to provide exploration. Using a convolutional layer will be beneficial in capturing the relations between different time steps of the data. Input dimension can be changed to (batch_size,105,5) and (batch_size,40,5) to use a convolutional layer. Further parameter tuning could improve all our models. Our lack of expertise in deep learning and long training times prevented us from optimizing our models to their full potentials.

References

Dai Yuqin, Wang Chris, Wang Iris, Xu Yilun. “Reinforcement Learning for FX trading”. Stanford University, Santa Clara, California, United States of America, 2019, Accessed: 2021 Available: https://stanford.edu/class/msande448/2019/Final_reports/gr2.pdf

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven Exploration by Self-supervised Prediction,” *arXiv.org*, 15-May-2017. [Online]. Available: https://arxiv.org/abs/1705.05363. [Accessed: 15-May-2021].