An Application of Proximal Policy Optimization to Algorithmic Trading

Alessandro Pavesi
Univeristy of Bologna
Bologna, Italy
`alessandro.pavesi2@studio.unibo.it`

Abstract

The use of new algorithms to approach older problems is one of the best methods to compare the results and to find if the newly proposed algorithms could work well in different domains. I approached the research made by Théate *et al.*[1] in which they developed an environment *"to solve the algorithmic trading problem of determining the optimal trading position at any point in time during a trading activity in the stock market"*; starting from that I wanted to try the Proximal Policy Optimization (PPO) presented by Schulman *et al.* and compare the results with the Trading Deep Q-Network (TDQN) used in the former paper.

1 Introduction

In this paper I will show the results about the comparison of TDQN and PPO, the PPO algorithm is based on Actor-Critic algorithm implemented over a Neural Network that I choose to be structurally identical to the one proposed on the TDQN. I will not formalize the problem of Algorithmic Trading nor the Proximal Policy Optimization algorithms because I will not capture every facets and my explanation would be approximate and incomplete, I invite the reader to check the original papers defined into the References section to retrieve the correct and complete information about them.

2 Problem Statement

2.1 Algorithmic Trading Environment

The Environment developed by Théate *et al.* is based on OpenAI Gym, so it exposes the usual functions *step, reset, and render*, and can be used as a normal Reinforcement Learning environment. The authors also implemented a Simulator that take care of loading the data and the agent we choose, and start the simulation in which each step uses the daily data given by the dataset instead of running an online environment. The Simulator includes various methods with which is possible to try an Agent, or Strategy, over a defined dataset, with the possibility to train a newly created Agent, to use an already trained Agent on different stock, and different methods to plot the results.

Dataset Usually a RL environment uses an interactive application underneath for taking the information to give the agent, in this case the authors decided to use dataset easily findable online, these datasets includes historic data about different market stock, divided in stock about companies like Sony, Apple or Tesla, and indices like S&P 500, Dow Jones or NASDAQ 100. The datasets includes daily data about the price from the first transaction of a trading day and the last one, the higher price during the day and the number of the units traded in that day. The range of data is equal for each market stock, from the start of 2012 to the end of 2019.

To correctly evaluate the agent the datasets are divided in training set and test set:

- **Training set:** $01/01/2017 \rightarrow 31/12/2017$.
- **Training set:** $01/01/2017 \rightarrow 31/12/2017$.

Observation Space The observation space includes the information just mentioned plus the state information about the RL agent that includes the current trading position, the number of shares owned and the available cash. To help the agent the data are composed not only by the current observation space just described but includes a series of the previous $\tau + 1$ daily data.

2.2 Neural Network

The neural network at the base of the Actor Critic algorithm for the PPO agent is inherited from the TDQN agent developed in the original paper. It's defined by three blocks composed as: a Linear transformation, a Layer Normalization, then a Leaky ReLU and finally a Dropout. After these three blocks there is the last Linear layer that define the number of output neurons on which, only in case of the Actor, is applied a softmax. So the architecture of the Actor and the Critic are the same, without consider the last Linear and the softmax, and they don't share any weights. The optimizer used is Adam and the loss is the MSELoss. Given that the AC algorithm is a Temporal Difference algorithm I optimize the training using a Buffer to keep track of the tuple needed.

3 Results and Discussion

Initially to check the correct functioning of the environment I choose to control the TDQN performance and check it with the results from the original paper. In that case the authors compare the results with some of the basic strategy that a simple Agent can use to operate into the stock market: Buy and Hold (B&H), Sell and Hold (S&H), Trend following with moving averages (TF) and Mean reversion with moving averages (MR). I will not discuss and show them instead I refer to the original paper for the explanation. After the first comparison to check the reproducibility of the results obtained by the authors, then I compare the results with the PPO algorithm implemented by myself, using different configuration of the network and different epochs of training.

All the code and the figures are available at link: https://github.com/pavva94/PPOAlgorithmicTrading

The performance are assessed using the quantitative performance assessment that consist in defining one performance indicator to numerically quantify the performance of a trading strategy. The authors used more than one indicators but the most important is the Sharpe Ratio and I decided to use it consequently. The Sharpe Ratio is widely used in the field of algorithmic trading because it combines both profitability and risk.

3.1 Reproducibility of original results

In this paragraph it's shown a comparison with the results of the original paper (called TDQN Original in the table) and the results given from a training of the TDQN executed by myself. The results are different and in this case are worst than the original but shows that the approach of the authors produce, in the average, a positive Sharpe Ratio.

Table 1: TDQN Results

	Sharpe Ratio	
Stock	TDNQ Original	TDQN
Tencent (0700.HK)	-0.198	-0.536
Apple (AAPL)	1.424	-1.067
AB InBev (ABI.BR)	0.187	0.555
Amazon (AMZN)	0.419	0.624
Alibaba (BABA)	0.021	0.225
Baidu (BIDU)	0.080	0.863
CCB (0939.HK)	0.202	-0.715
Coca Cola (KO)	1.068	1.031
Dow Jones (DIA)	0.684	0.684
Nikkei 100 (EWJ)	0.019	-0.312
FTSE 100 (EZU)	0.103	0.128
Facebook (FB)	0.151	0.269
Google (GOOGL)	0.227	-0.269
HSBC (HSBC)	0.011	-0.297
JP Morgan (JPM)	0.722	0.713
Kirin (2503.T)	0.852	1.581
Microfsoft (MSFT)	0.987	1.511
Nokia (NOK)	-0.094	-0.563
Philips (PHIA.AS)	0.675	-0.200
PetroChina (PTR)	0.156	-0.695
NASDAQ 100 (QQQ)	0.845	0.260
Shell (RDSA.AS)	0.425	0.569
Siemens (SIE.DE)	0.426	0.201
Sony (6758.T)	0.424	0.011
S&P 500 (SPY)	0.834	0.834
Toyota (7203.T)	0.304	-0.339
Tesla (TSL)	0.621	0.260
Twitter (TWRT)	0.238	-0.310
Volkswagen (VOW3.DE)	0.216	-0.178
ExxonMobil (XOM)	0.098	0.055
Average	0,404	0,163

3.2 PPO variants

Here the results of the PPO algorithm applied to the Algorithmic Trading problem, comparing different neural network and training strategies. In particular the first three columns shows the results of different size of neural network; the first one tested uses the same network of the original paper without modifications, the second test is made using a network with half the size of the original one, and the third test uses a network with double the size of the original one. Finally the last column shows the results of the original network but trained with more training epochs, precisely double. As it's shown the best results are retrieved by the network in the original form, confirming the good work of the author in seeking a network that can work properly. Another important point is that increasing the number of epochs doesn't implies an improvement of the performance, at least in this case.

Table 2: PPO variants

	Sharpe Ratio			
Stock	PPO Standard	PPO Small	PPO Big	PPO Long
Tencent (0700.HK)	0.777	-0.123	0.309	-1.579
Apple (AAPL)	0.158	-1.669	0.960	-0.125
AB InBev (ABI.BR)	0.196	-0.643	-0.694	-0.147
Amazon (AMZN)	0.284	-0.609	-0.967	-0.786
Alibaba (BABA)	-0.234	0.357	-1.196	0.002
Baidu (BIDU)	-1.182	0.289	-1.074	-1.481
CCB (0939.HK)	-1.014	-0.816	-1.094	-1.257
Coca Cola (KO)	-0.234	0.657	-0.359	0.063
Dow Jones (DIA)	1.013	-1.169	0.611	-0.827
Nikkei 100 (EWJ)	-0.910	-0.893	-1.020	0.183
FTSE 100 (EZU)	0.082	-0.849	0.864	-2.114
Facebook (FB)	-0.146	1.210	0.129	0.408
Google (GOOGL)	-1.798	-1.399	-0.775	-1.092
HSBC (HSBC)	-0.035	-1.290	-1.524	-1.154
JP Morgan (JPM)	-0.824	0.183	-0.491	-2.010
Kirin (2503.T)	1.278	-0.176	0.406	0.406
Microfsoft (MSFT)	1.266	-0.628	-0.628	1.303
Nokia (NOK)	-0.225	-0.563	-1.057	1.185
Philips (PHIA.AS)	-0.833	-1.282	-1.619	-1.254
PetroChina (PTR)	-0.381	-0.645	0.921	-1.993
NASDAQ 100 (QQQ)	-0.100	-0.735	0.566	0.016
Shell (RDSA.AS)	-1.318	-0.557	-0.616	-1.408
Siemens (SIE.DE)	-2.023	-1.005	-1.194	-0.525
Sony (6758.T)	-0.123	-2.189	0.494	-0.047
S&P 500 (SPY)	-0.871	0.162	0.538	-0.252
Toyota (7203.T)	-0.897	0.140	0.262	0.255
Tesla (TSL)	-0.263	-1.021	-0.324	-0.118
Twitter (TWRT)	0.827	0.325	0.655	-0.039
Volkswagen (VOW3.DE)	-0.711	-0.461	-0.493	-0.083
ExxonMobil (XOM)	-0.070	-0.094	-0.611	-0.460
Average	-0,277	-0,516	-0,301	-0,497

3.3 PPO vs TDQN

To show the results obtained here it's shown a table comparing the results of the PPO and TDQN algorithms both trained by me.

Table 3: PPO vs TDQN

	Sharpe Ratio	
Stock	PPO	TDQN
Tencent (0700.HK)	0.777	-0.536
Apple (AAPL)	0.158	-1.067
AB InBev (ABI.BR)	0.196	0.555
Amazon (AMZN)	0.284	0.624
Alibaba (BABA)	-0.234	0.225
Baidu (BIDU)	-1.182	0.863
CCB (0939.HK)	-1.014	-0.715
Coca Cola (KO)	-0.234	1.031
Dow Jones (DIA)	1.013	0.684
Nikkei 100 (EWJ)	-0.910	-0.312
FTSE 100 (EZU)	0.082	0.128
Facebook (FB)	-0.146	0.269
Google (GOOGL)	-1.798	-0.269
HSBC (HSBC)	-0.035	-0.297
JP Morgan (JPM)	-0.824	0.713
Kirin (2503.T)	1.278	1.581
Microfsoft (MSFT)	1.266	1.511
Nokia (NOK)	-0.225	-0.563
Philips (PHIA.AS)	-0.833	-0.200
PetroChina (PTR)	-0.381	-0.695
NASDAQ 100 (QQQ)	-0.100	0.260
Shell (RDSA.AS)	-1.318	0.569
Siemens (SIE.DE)	-2.023	0.201
Sony (6758.T)	-0.123	0.011
S&P 500 (SPY)	-0.871	0.834
Toyota (7203.T)	-0.897	-0.339
Tesla (TSL)	-0.263	0.260
Twitter (TWRT)	0.827	-0.310
Volkswagen (VOW3.DE)	-0.711	-0.178
ExxonMobil (XOM)	-0.070	0.055
Average	-0,277	0,163

3.4 Rendering

Thanks to the environment created by the authors it's possible to see how the agent act during the simulation. The figures below shown when the agent sell and buy the actions of the stock, compared to the capital owned by the agent at the same time so it's possible to see how the finances of the agent change during the simulation. In these simulation the Agent's methodology of operating is to sell and buy nearly at each day. Below only a simulation over the Apple and Nokia stocks are shown for brevity.

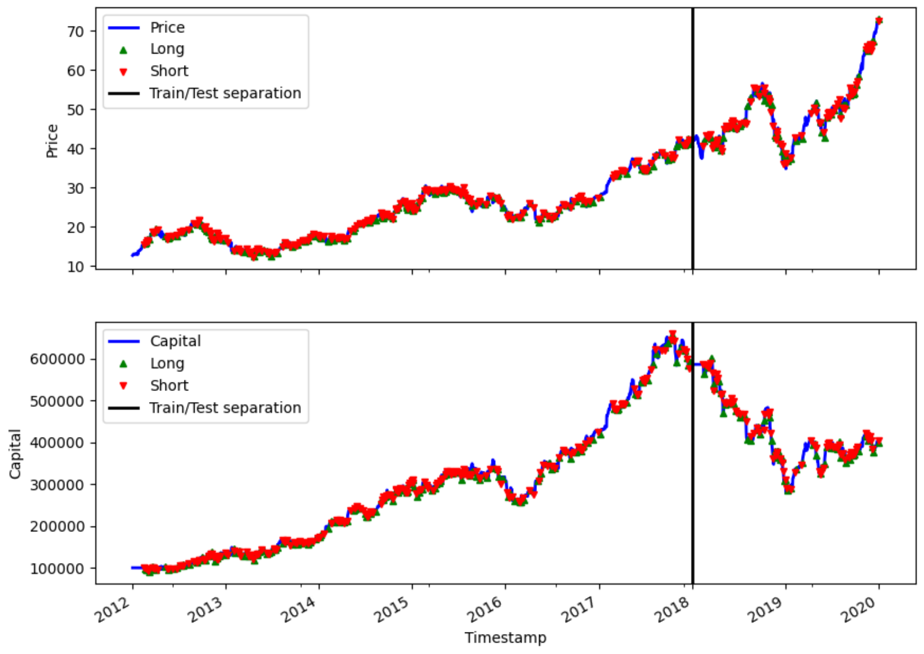


Figure 1: Apple Simulation

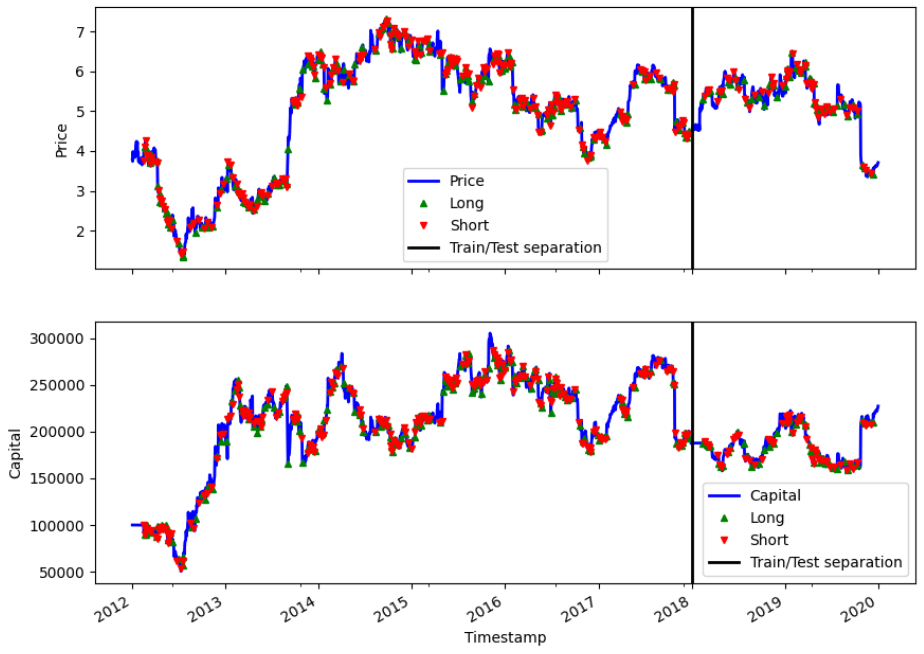
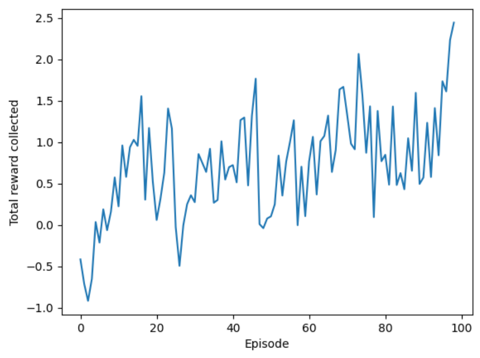
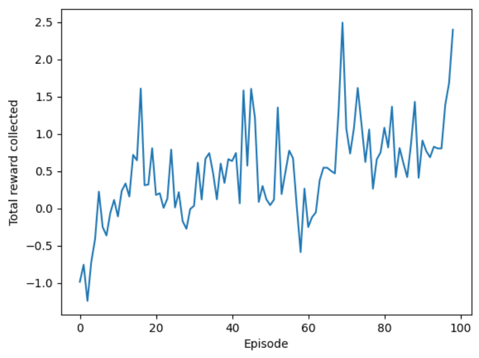


Figure 2: Nokia Simulation

Rewards Another plot useful to check how the agent learn is the total reward. During the training the variance is high but the trend is increasing so the agent are learning how to gain more reward.



(a) Apple Rewards



(b) Nokia Rewards

4 Conclusions

The scope of this research was to apply the Proximal Policy Optimization and compare it with the Trading Deep Q-Network proposed by Théate *et al.* over the problem of Algorithmic Trading. The application of PPO raises the complexity of the implementation given the complex structure of the PPO Algorithm but shows it can be applied to the stock market and check the agent can learn how to act in the market. The results are worse compared to the TDQN, of course the tuning made by the original authors on the DQN to create the Trading DQN are way better than what I've made here in which I used theirs network into the Actor Critic part. Moreover the parameter I've used in the PPO Agent are the tested and recommended by the researchers that proposed the PPO Algorithm. The results can be seen as positive, the PPO agent learns how to act though the average performance brings the agent to lose money from the initial capital. A more accurate tuning of the parameter given a study over it maybe with an ensemble technique or different technique of training can increase the performance.

References

[1] T. Théate, & D. Ernst, (2020) *An Application of Deep Reinforcement Learning to Algorithmic Trading*, arXiv

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, (2017) *Proximal Policy Optimization Algorithms*, arXiv