Deep Q-learning for a trading agent using a financial time series

Marc Dabad Planas

Universitat Ramon Llull - La Salle

Abstract

Reinforcement learning has been a widely explored technique to create agents that respond optimally in a time-series environment. This paper has been created to explore the limits of this technique using real-world data for a time series in the crypto-market. Xiang Gao in his paper *Deep reinforcement learning for time series: playing idealized trading games* [1], used Deep Q-learning to create an optimal policy for an agent with made-up procedural data and its results are promising. So it has been used as a benchmark to compare with similar architectures but using real data from the BTC/USDT pair in the cryptomarket. The results of the experiment show that using the same approach as X. Gao, but with real data ended up with 66.6% success using a MLP architecture.

Keywords: *Reinforcement Learning, Q-learning, Deep Learning, Trading Agent, MLP, LSTM, Time Series, Bitcoin.*

1. Introduction

Reinforcement learning emerged from the combination of psychology, artificial intelligence, and optimization theory. It was first introduced by American psychologist B. F. Skinner in the 1940s and published its findings in his article *Superstition in the Pigeon*, in 1948. [2] Skinner conducted a series of experiments in which he controlled the behavior of pigeons by reinforcing desired actions to create a defined behavior. Skinner established a precedent from which future research could base its investigations to define a process that uses rewards to shape a particular behavior. In particular, in the 1980s, American computer scientist Richard S. Sutton and British artificial intelligence professor Andrew G. Barto combined these ideas and proposed the Q-learning algorithm, which is one of the most widely used reinforcement learning algorithms. Q-learning allows an agent to learn how to take optimal actions in an unknown environment by using a Q-value function, which assigns a reward to every possible state-action. Q-learning has been successfully used in a wide variety of problems, including robot control, video game design, and data analysis. They published their research in their book *Reinforcement Learning: An Introduction* [3] in 1998, which has served as the fundamental theory that current reinforcement learning projects are based on. The Q-learning technique helps to define a process in which the action taken by an agent depends on future actions and states, so it can be applied to finance, healthcare and other industries, something hard to do with conventional supervised learning methods.

Some of the remarkable projects is the one made by *Silver et al* [4] that showed amazing results in the field of *Deep Reinforcement Learning*, by creating an agent, named *Alpha Go*, that plays the legendary *Go* table game in real time by just using raw data.

And surpassed the *Go* world champion, Lee Sedol [5]. These kinds of projects showed the potential of creating a Reinforcement Learning (RL) Agent to take optimal actions in a stochastic process like time series [6].

It's worth mentioning the fact that recurrent neural networks (RNN) are commonly used for the prediction of events in time series. As shown by Lipton *et al* [7] who used Long-Short Term Memory (LSTM) layers to classify successfully multivariate time series in the clinical measurement field.

It is in 2018, when X. Gao, translated the success of RL agents in time series environments to the financial field, by using the simplest approach of generating artificial inputs to train and evaluate de RL agent in order to maximize an expected return.

Naturally, it is obvious to wonder if the success that X. Gao obtained in his project can be repeated using real data such as the one obtained by the pair *BTC/USDT* that represents the price of the cryptocurrency Bitcoin based on the value of the US dollar. And this is the question that this paper is addressing.

2. Related projects and Hypothesis

2.1 X. Gao - *Deep reinforcement learning for time series: playing idealized trading games*

In Gao's project, the agent receives 40 samples of the time series describing the price at each instant in time and then chooses an action that results in a reward.

The agent has three actions available that it can take depending on the situation in which it is. The three actions are CASH (do not buy the asset, or sell the asset if you have bought it), BUY (buy the asset), HOLD (do not sell the asset purchased).

Therefore, the actions could be represented as follows:

$$\text{valid actions} = \begin{cases} \{\text{CASH, BUY}\} & \text{not holding any stock} \\ \{\text{CASH, HOLD}\} & \text{otherwise} \end{cases}$$

Figure 1: Actions that can be performed by the agent.

In the project created, the same actions are implemented for the agent. As well as the following rewards per action that Gao implemented:

$$r_t = \begin{cases} 0 & \text{CASH} \\ p_{t+1} - p_t - c & \text{BUY} \\ p_{t+1} - p_t & \text{HOLD} \end{cases}$$

Figure 2: Rewards given to the agent for each action.

With this mapping, Gao intends for the agent to be rewarded positively if it buys an asset that will increase in price, and vice versa. Similarly, if the agent has bought an asset and holds it (without selling), and it increases in price, the reward will also be positive, but negative if it decreases.

Gao used different architectures to test its RL agent. Some of them are a MLP and a LSTM combined with dense layers. Both architectures are replicated in the project presented in this paper.

The results of the experiment are promising as the model regardless of the architecture used manages to get a positive number of accumulated rewards:

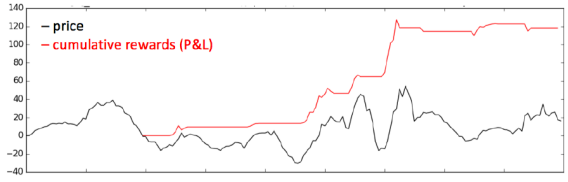

Figure 3: A test episode of the Univariate game played by the trained agent.

In particular, with test data the percentage of profitable episodes performed by the agent are:

Architecture	Profitable Episodes (Out-of-Sample)
MLP	97%
LSTM	94%

Table 1: Profitability of the agent with different architectures after 100 episodes.

These results are really promising and look like they could be translated to a project that uses real world data.

2.2 Hypothesis

As said, it's natural to follow the same approach as X. Gao, but with real data, and as his project, this one uses the *Buy&Hold* strategy to compare the benefits that the agent gets with the strategy created from the chosen model. It is opportune to use the Buy&Hold strategy, since it will follow the trend and evolution of the asset price, and this allows to know how difficult it is to make profits in the proposed scenario, because it is not the same to make money from buying and selling when the asset has an increasing trend, than when the trend is decreasing.

Therefore, a theoretical hypothesis could be defined such that: "An agent with real data from *BTC/USDT* who uses deep Q-learning as a methodology for estimating accumulated earnings is able to invest money beneficially and even exceed the benefits of a simple strategy like the *Buy&Hold* strategy."

To test the hypothesis and challenge it properly, 3 scenarios have been created in which each one has its particularities, because of the different kinds of fluctuations of the time series. This way the hypothesis can be accepted or rejected depending on the scenario.

3. Tech Stack

3.1 Docker

In order to facilitate the export of the project a Docker [8] image has been created that uses as base the *python.3.9-slim* image and adds the requirements necessary to run the agent. This way there's an isolation of the dependencies of the project.

3.2 Programming Language

The agent implementation is carried out using the Python programming language which has libraries such as Numpy or Pandas that facilitate the implementation of data management (open source language created in the late 1980s by Guido van Rossum) [9].

Python is commonly used in Data Science projects because it is easy to learn and has a wide range of libraries that can be used for data analysis, manipulation and visualization.

3.3 Python's Libraries

The libraries used to implement the RL agent are Pytorch for the Deep Learning models, Pandas and Numpy to facilitate the data manipulation, requests to obtain the data from the API and matplotlib to create the visualizations and store them in the volume associated with the docker container.

4. Experiment

AS mentioned 3 scenarios have been created, each one with its own real data from the historic *BTC/USDT*, and all of them with 1000 hours of time series. At the beginning of the evaluation the agent is provided with $1000 which can be invested by using the learnt policy during training.

4.1. Agent's Training

For the agent's training 7000 hours of data has been used. During this series, the price of *BTC/USDT* evolves in different ways, which is of use by the agent in order to train in different scenarios.

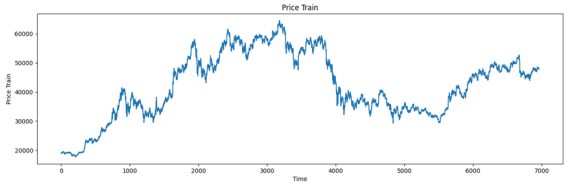

Figure 4: Time series used to train the RL agent.

As can be seen, the price not only decreases, increases or goes sideways, but it has all of them in the same dataset.

The training has been completed after 3 episodes, since it showed no improvement after this number of episodes. The other parameters used are a learning rate of 0.00024 and a gamma of 0.8, which is defined in the *Bellman Equation* [10] to calculate the Target Q-values.

4.2. Scenario 1: Mixed Fluctuations

The first test scenario consists of different opportunities for the agent, since there are certain increases in the asset, there is also certain sideways movement and decreases in the price of the asset, so it can be said that it is a mixed scenario. See below the dataset used for this scenario:

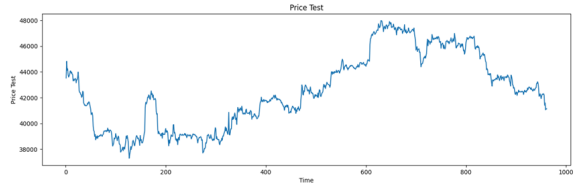

Figure 5: Time series used for scenario 1.

As can be seen, there is a clear drop in price at the beginning, then there's a huge opportunity to get some profitable actions, it continues with an increasing trend and ends up with a negative trend.

Figure 6: Money evolution and actions made by the agent with a MLP model in scenario 1.

By using both the MLP and LSTM architecture the results have been positive, considering that the agent started with $1000 and ended up with $1136.56 for the MLP (Figure above) and $1045.52 for the LSTM. Also if compared to the *Buy&Hold* strategy the outcome would be of $917.93 which is below the agent's strategy.

By performing a Wilcoxon [11] test for both strategies with a significance level of 0.05 and considering the following definition:

- Null Hypothesis (H_0): The mean of both distributions are equal.
- Alternative Hypothesis (H_1):The agent's strategy increases the mean value of the benefits.

The results showed a p-value of 8.07e-159, so H_1 has been accepted.

It's worth mentioning that the number of profitable operations has been 60% for the MLP and 50% for the LSTM.

4.3. Scenario 2: Positive Trend

In this scenario, the trend is clearly rising, which should result in an increase in the profits that the agent can achieve. This scenario has been chosen so it can be seen if the agent is able to ideally buy at the beginning, maintain the position and then sell at the highest point on the chart.

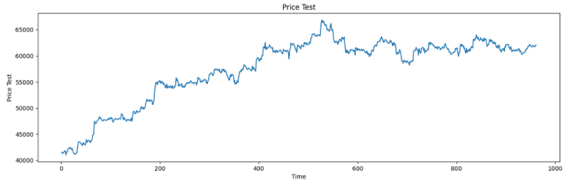

Figure 7: Time series used for scenario 2.

Unfortunately, the results haven't been as positive as in the first scenario.

Figure 8: Money evolution and actions made by the agent with a MLP model in scenario 2.

For the Buy&Hold strategy, the outcome would be $1493.42. On the other hand, the outcome of the agent's strategy has been $1430.44 and $1085.23 for the MLP (Figure above) and the LSTM respectively.

By performing the Wilcoxon test in this scenario the p-value resulted in 1.0, which is obviously higher than the significance level. So the Null Hypothesis has been accepted.

4.4. Scenario 3: Negative Trend

The last scenario is the one in which it is more difficult to make profits since it mostly has a declining trend, but at the same time with the Buy&Hold strategy, the investment balance would be negative, so it is interesting to compare both strategies in this type of circumstances. See the evolution of the price of the asset in this case:

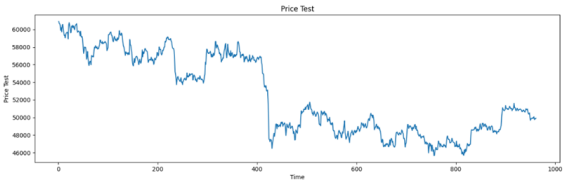

Figure 9: Time series used for scenario 3.

The agent in this scenario showed that it can write off losses due to price drops, since the resulting capital of the agent's investment strategy has been $896.37 for the MLP and $934.64 for the LSTM.

Check out the evolution of the agent's strategy with the MLP architecture:

Figure 10: Money evolution and actions made by the agent with a MLP model in scenario 3.

Both benefits achieved by the agent are higher than the Buy&Hold strategy which results in $819.37.

After executing the Wilcoxon test with the same level of significance (0.05) the Alternative Hypothesis (H_1) has been accepted for both architectures since the p-value is 2.78e-101 (MLP), and 8.15e-159 (LSTM).

If the number of profitable operations is observed, the percentage is 40% for the MLP and 47.8% for the LSTM.

5. Results

The agent had in total a percentage of 61.1% of profitable operations with the MLP model and 56.5% with the LSTM model. Also the amount of benefits obtained are different depending on the architecture, resulting higher with the MLP in scenario 1 ($1136.56) and 2 ($1430.44) and lower in scenario 3, since the LSTM accumulated $934.64 in total.

As a summary of the hypothesis test results the following table can be found:

	MLP	LSTM
SIT1 - Mixed	H1 Acc.	H1 Acc.
SIT2 - Positive	H0 Acc.	H0 Acc.
SIT3 - Negative	H1 Acc.	H1 Acc.

Table 2: Hypothesis test result for each scenario.

Both architectures showed similar results regarding the hypothesis test, but at different degrees, since the amount of benefits obtained by each model is different in every scenario.

6. Discussion

It has been seen that the results compared to Gao's have some difference which can lead to a theory that would explain such a difference and it's that the artificially generated data has an intrinsic pattern that the agent can subtract by using the models. On the other hand, the price of *BTC/USDT* has some fluctuations that aren't defined in the time series itself but that is influenced by other variables.

Seems that RL is a powerful tool to create agents that learn an optimal policy, but only with the right data provided. So probably by adding some other indicators as extra data for the agent the results could be improved.

7. Conclusions and future works

Although the hypothesis has been proved in 2 of the 3 scenarios, it's not proper to say that the agent's strategy is always better than the Buy&Hold one. The agent doesn't anticipate the price fluctuation but it understands the trends in order to take profit. So in the rising trend scenario (2) the agent makes some profit from the increasing value of the price, but it doesn't sell when the price is at its peak, because of its lack of anticipation. Instead in a negative trend (scenario 3) it does take a few opportunities to be profitable which makes the difference with the Buy&Hold strategy.

Also there has been some significant difference in the performance between architectures, resulting in higher incomes for the MLP model in the rising trend scenario and the mixed scenario, but lower than the LSTM in the negative trend scenario. More tests with different scenarios should be done in order to confirm that the LSTM model does a better job in scenarios with low opportunities to make profitable trades.

An important point is that the data used is only the price of the asset over time. This decision was made because the goal was to create a very simple agent without any other type of data or indicator. Therefore, one could think that the agent's strategy could be improved if the other data is provided, such as the prices of High or Low [12] or the popular trading indicators like the RSI [13] or the MACD [14].

The future of the project includes exploring more recurrent architectures such as the GRU [15], as well as convolutional networks, or even modifying the already created architectures (MLP and LSTM) to test a different number of layers or units per layer to see if results can be improved.

8. Referencias

[1] (X. Gao, 2018) *Deep reinforcement learning for time series: playing idealized trading games,* enlace.

[2] (B.F. Skinner, 1948) *Superstition in the Pigeon*, enlace.

[3] (R. Sutton y A. Barto, 1998) *Reinforcement Learning: An Introduction*, enlace.

[4] (D. Silver et al, 1998) *Mastering the game of Go with deep neural networks and tree search*, enlace.

[5] (Wikipedia, 2022) *Lee Sedol*, enlace.

[6] (Wikipedia, 2022) *Stochastic Process*, enlace.

[7] (ZC. Lipton et al, 2015) *Learning to Diagnose with LSTM Recurrent Neural Networks*, enlace.

[8] (Docker Inc, 2022) *Docker's Webpage*, enlace.

[9] (G. van Rossum, 1995) *Python's Reference Manual*, enlace.

[10] (R. Bellman, 1954) *The Theory of Dynamic Programming*, enlace.

[11] (Adam Hayes, 2021) *Wilcoxon Test*, enlace.

[12] (Cory Mitchell, 2022) *Understanding Basic Candlestick Charts*, enlace.

[13] (Jason Fernando, 2022) *Relative Strength Index*, enlace.

[14] (Jason Fernando, 2022) *Moving Average Convergence Divergence*, enlace.

[15] (Wikipedia, 2022) *Gated Recurrent Unit*, enlace.