REINFORCEMENT LEARNING PROJECT

FUTURE STOCK PREDICTION AND TRADING USING ACTOR-CRITIC METHOD

Saivinay Goriparthi

2019AAPS0255H

Challa Sai Reshwanth

2019A3PS0433H

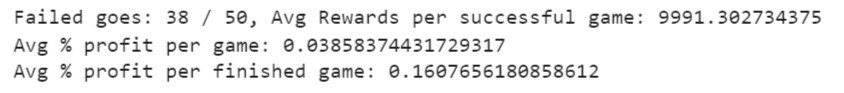
**Paper Name:**

*“A blundering guide to making a deep actor-critic bot for stock trading”* - Tom Grek

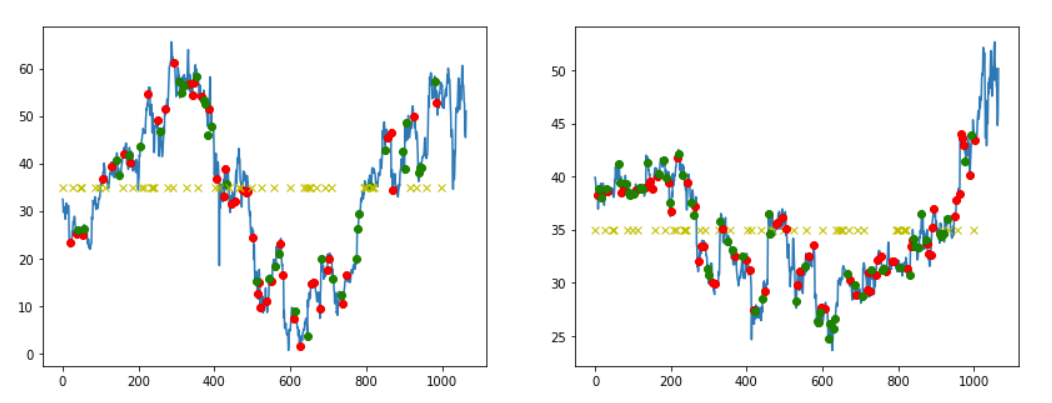
**Link to Project:**

<https://colab.research.google.com/drive/1ZmtI3As9R1ef1cA50eUHHqYoqQmzSJwj?usp=sharing>

**Our Results:**



(The starting portfolio value was on average about 9,600, so 10,000 rewards represent a 4% rise.) The bot learned the rules, too: it didn’t bankrupt itself or try to sell more shares than it had, except one time. Here’s how it looked on a section of training data:



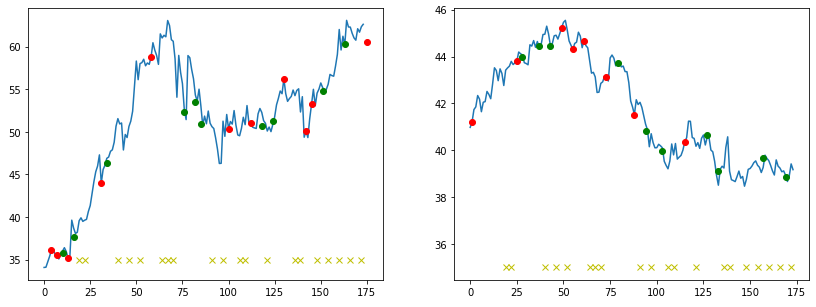
Here’s one of its “trading strategies”, graphed. Red means buy, green means sell, and left is AAPL and right MSFT. The yellow crosses are timesteps where the bot chose to take no action.

**Results From the Paper:**

The bot learned to not cheat or bankrupt itself, most of the time, and it was able to make a significant profit: here you can see 36% returns (on the data it trained on).



(Here, 0.36 means 36% profit)

(The starting portfolio value was on average about 3000, so 4000 rewards represent a 36% rise.) The bot learned the rules, too: it didn’t bankrupt itself or try to sell more shares than it had, except one time. Here’s how it looked on a section of training data: (Red: BUY, Green: SELL, Yellow Cross: do nothing)

**Comparison between the results:**

|  | Our result | Result in the paper |
| --- | --- | --- |
| Probability of Failure | 0.76 | 0.02 |
| Avg % profit per game | 0.0385837 | 0.35716032 |
| Avg % profit per finished game | 0.1607656 | 0.36444932 |

We Could observe that there is a more probability of failure in our result compared to the result in the paper this is due to the fact that we have chosen a different time period compared to that in the given paper and we have chosen a smaller time frame which has resulted in the Avg % profit per game and the Avg % profit per finished game to be less than that of the paper.

**Introduction of the new idea:**

The important learning is that if different states yield the same reward, it complicates training and slows convergence. So the useful trick is:

*Add a time element to the reward. As time moves on, actions that don’t kill the game are successively more valuable; or, the longer the agent continues in its environment, the more reward it gets.*

We were giving the agent a small positive reward for actions and a small negative reward for doing nothing. So, an action like “buy AAPL” might yield +0.1 reward at any time during the sequence.

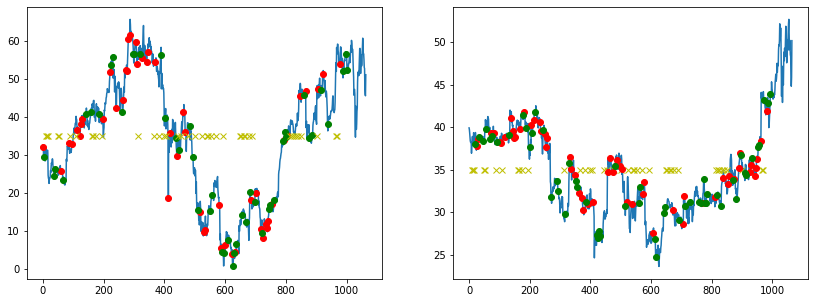
But we ultimately want the bot to keep going til the end of the sequence without bankrupting itself, which I encouraged by mixing in a time element, namely, subtracting the number of steps left to go from the reward. “buy AAPL” with 50 steps to go might yield a reward of -49.9; with 49 steps to the reward would be -48.9 (better), and so on.

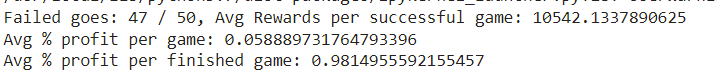
Some of the advantages of using this idea:

Compensate for the infrequency of good rewards by increasing them a lot compared to all the negative rewards. There’s no need to subtract the starting portfolio value from the ending value when calculating a reward. You get this for free as the bot always tries to optimize higher.

**Our results after implementation of the new idea:**

(The starting portfolio value was on average about 10,000, so 10,550 rewards represent a 6% rise.) The bot learned the rules, too: it didn’t bankrupt itself or try to sell more shares than it had, except one time. Here’s how it looked on a section of training data:





**Comparison between the results:**

|  | Our result | Result after idea implementation |
| --- | --- | --- |
| Probability of Failure | 0.76 | 0.94 |
| Avg % profit per game | 0.0385837 | 0.05888973 |
| Avg % profit per finished game | 0.1607656 | 0.98149555 |

We could see from the results that after implementation of the idea the Avg % profit per game is increased from 4% to 6% as it is able to train on the given dataset more efficiently due to the introduction of a new time-varying reward element which is helping it to maximize its gain but it is not able to survive longer.