SoC Summer 2021 Final Documentation

**** This will be considered as your final documentation and will be made available for online usage.***

R(ea)L Trader

Pratyush Aggarwal, Yash Gupta, Raaghav Raaj

Keywords

Stock Trading, ML, Reinforcement Learning, Strategies, Optimizing Profit, S & P 500, NSE

Team Member Name	Roll Number	Email-Id
Shubh Kumar	200050134	shubh5796@gmail.com

Problem Statement

As a kid, you were always given a reward for excelling in sports or studies. Also, you were reprimanded or scolded for doing something mischievous like breaking a vase. This was a way to change your behaviour. Suppose you would get a bicycle or PlayStation for coming first, you would practice a lot to come first. And since you knew that breaking a vase meant trouble, you would be careful around it. This is called reinforcement learning.

The reward served as positive reinforcement while the punishment served as negative reinforcement. In this manner, your elders shaped your learning. In a similar way, the RL algorithm can learn to trade in financial markets on its own by looking at the rewards or punishments received for the actions.

In the realm of trading, the problem can be stated in multiple ways such as to maximise profit, reduce drawdowns, or portfolio allocation. The RL algorithm will learn the strategy to maximise long-term rewards.

Existing solutions in the Market

There are many companies which actively participate in Automated Stock Trading and make billions out of it. They have complex strategies ensembling both RL and non-RL based models, we tried to only look at how a part of it would perform while automating trading on a micro-scale.

Proposed Solution

We (I) completed two fully-functional models, one of which uses Deep-Deterministic(not exactly) Policy Gradient to come up with an Optimal Q-Value prediction using a DNN, and this is then used to dynamically calculate the Optimal Action. This was a slight variation of the actual DDPG model.

The Second Real-Time Automated Model had a static RL Model and a dynamically updated Return and Max. Drawdown Predicting model, these two would together do the trading.

Brief Description

The Project started by learning about RL, Markov Chains and the basics of Finance (Optimizing Portfolios, going over basic strategies like Momentum, Pairs Trading, Short Term Mean Reversal etc.). In order to test these skills in action, I built a simple Grid Solver which calculates Q-values and follows the path which maximises its cumulative, and also tested these strategies on QuantInsti.com (This code was mostly adapted and hence, not available!)

It was originally planned that we would use any Online Platform(BlueShift or QuantInsti) for our models, but I later saw that even after using OpenAI Gym, it couldn't be sped up enough on these platforms, hence, I finally decided to develop the entire BackTesting Procedure from Scratch as well! Due to this, we weren't able to participate in Competitions online (which was also a checkpoint!)

The above two models, like that, were implemented pretty well.

Progress

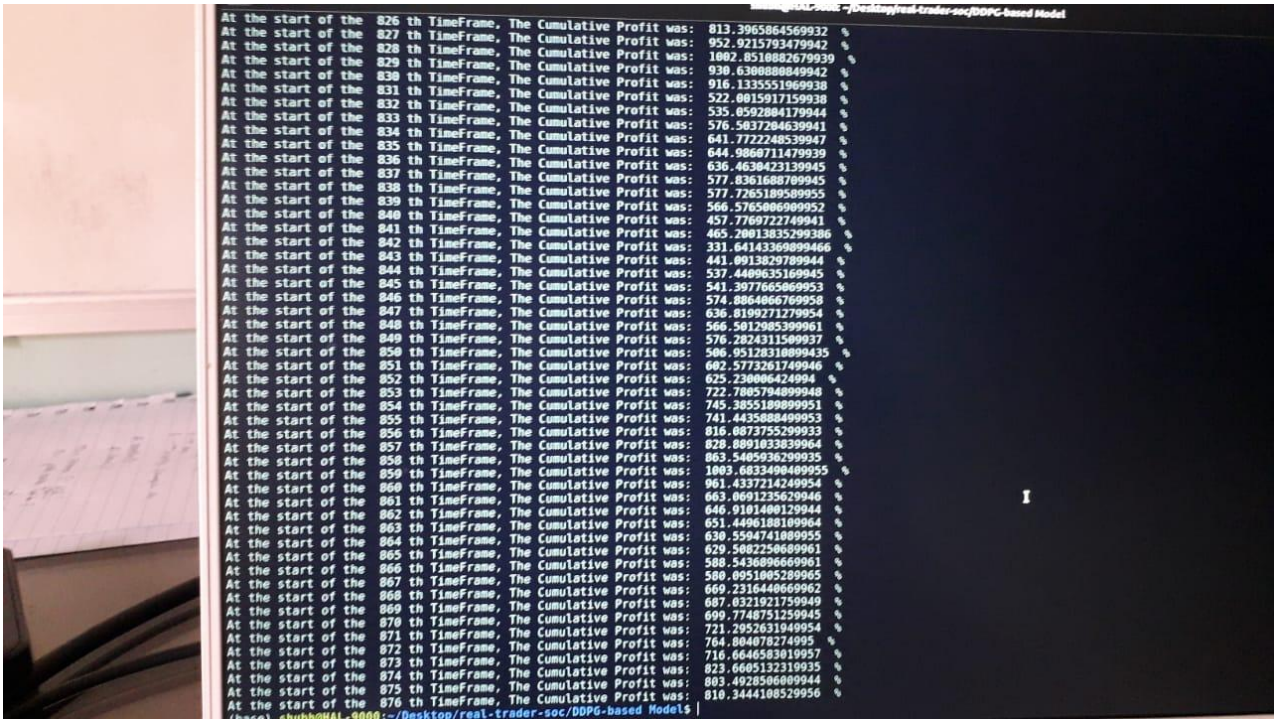
As I completed the Project alone, I would go on and just describe the various parts and till when I was able to complete these :

1. Reading up about RL, Markov Chains (finished by 31st April)
    a. Markov Decision Processes
    b. Bellman Equation and Optimality
    c. Using Dynamic Programming to find optimal solutions to Bellman's Equation
2. Using whatever we learned about RL in order to build a basic Grid Solver (Completed around 10 May)
3. Learning about Finance, Stock Markets and various Strategies used there. (completed by 30th of May)
    a. Virtual Stock Trading on Invstr App (even placed 3rd globally for beginner's contest)
    b. Using Python for Financial Applications (Learning about modelling portfolios, Confidence Intervals in Python and basic algorithms to optimize them!)
    c. Learning about basic strategies used in Trading like Momentum, Pairs Trading and Short-Term Mean Reversal etc.

4. Implementing the first Model, a slight variation of DDPG, as its wasn't exactly deterministic, but it provided better results than the Deterministic one as shown in the paper(1). This was done with 30 S&P 500 stocks with data from 2009-2020. (Completed by 3rd of June)
5. Implemented a RealTime-Automated Stock Trading Algorithm with a static(just trained once!) RL Action Predictor and a dynamic(changed as we get new data about stocks) Return Predictor on NSE based stocks Tata Steel, Reliance Industries, Vodafone-Idea and Kotak Mahindra. This was done on minute-wise data obtained using a semi-automated Algorithm from MoneyControl.com (Completed by 13 July)

Results

The Grid Solver was able to solve any grid till 1000-by-1000 in no time, which one may check for oneself.
As for the first model, it delivered ~810% profit in just 3 years, against the 100% in 10 years' which the Author's Model delivered. This was so because my model had a non-deterministic action predictor and had no risk-aversion. The Profit even peaked at somewhere around 1400% in the middle.



The second RealTime Automated Model did just as well, but I wasn't able to test it well as High Frequency data isn't easy to get and the data I got from my semi-automated model had its errors!

Even then It did report ~125% profit in about 30 days of trading every minute (but that was on the Training Set!)

Learning Value

Reinforcement Learning in depth!
PPO and DDPG (two of the most popular methods in RL)
Stock Markets and Portfolios
Trading Strategies in depth, which I was then able to use in order to design WnCC's TSS-Python's Week-2 Assignment.
Building DNNs from Scratch with Adam Optimizer.

Software used

Python
Pinta
Numpy
Pandas
Tensorflow, Stable-baselines, and the application developed by BlueShift and QuantInsti were used for testing and learning about various Strategies.

Suggestions for others

Try to come up with better ways to get High-Frequency data using Web Scraping or other methods, as it's not easy to get your hands on that.

You should also be familiar with ML(at least) and Markov Decision Processes(if possible) before building any Stock Trading Model.

Playing around with Virtual Stock Trading games like Invstr often gives you enough exposure with Markets to make operational decisions while building your model.

Contribution by each Team Member

I was the lone-team member :')

References and Citations

(1) *Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy : https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3690996*

(2) *https://github.com/AI4Finance-LLC/Deep-Reinforcement-Learning-for-Automated-Stock-Trading-Ensemble-Strategy-ICAIF-2020 : This is the Model made by the authors of the above paper.*

Disclaimer

Fair use of the Data got from the above paper, from Github. All other things are made from Scratch. The NN Model is partly similar to Andrew Ng's Deep Learning Specialization.

Licenses

The Github Repository used was under MIT License.