Deep Trading Agent using DeepSense Network for Q function approximation

Manasa Veena Bollam
Department of Computer Science
Georgia State University
mbollam1@student.gsu.edu

Soni Gudur
Department of Computer Science
Georgia State University
sgudur1@student.gsu.edu

Abstract

The trading environment is modeled as a Markov decision process (MDP) in reinforcement learning for trading, where the trader makes decisions based on the state of the market and gets feedback in the form of rewards or penalties. The goal is to discover a strategy that maximizes the anticipated long-term benefit. The suggested fix involves using a DeepSense Network-based Trading Assistant that is based on Deep Reinforcement Learning to approximate the Q function.

The trading environment is modeled as a Markov decision process (MDP) in reinforcement learning for trading, where the trader makes decisions based on the state of the market and gets feedback in the form of rewards or penalties. The goal is to discover a strategy that maximizes the anticipated long-term benefit. The suggested fix involves using a DeepSense Network-based Trading Assistant that is based on Deep Reinforcement Learning to approximate the Q function.

Deep Q-Trading, which uses a simplified trading problem for a single asset to solve it, is the source of inspiration for Trading Model. Only one of the three possible actions—neutral(1), long(2), or short(3)— is permitted for each trade unit, and the reward depends on the agent's existing position. The goal of the deep learning agent is to maximize the total number of accumulated rewards. The Deep Sense architecture is used to modify the existing Deep Q-Trading model in order to approximate the Q function. Deep Q Learning, which estimates the values using a deep neural network. If the relative importance is maintained, this approximation of values is not harmful. Deep Q-fundamental Learning's working procedure entails feeding the starting state into the neural network, which then outputs the Q-value for each potential action. The reinforcement learning technique known as "Deep Q-Learning" uses a deep neural network to simulate the Q-function, which is used to decide the best course of action to take in each condition. The predicted cumulative reward of performing a specific activity in a specific condition and adhering to a specific policy is represented by the Q-function. The Q-function is iteratively modified in Q-Learning as the agent engages with the environment. Applications for Deep Q-Learning include robotics, autonomous cars, and gaming.

1 Progress

1.1 Requirements

Project requires the following requirements :

- Python
- pandas : For pre-processing Bitcoin price series
- tqdm: For displaying progress of training
- Tensorflow

1.2 Bitcoin Dataset

The current state of the agent is thought to be sufficient in the Markov scenario of reinforcement learning to determine the agent's subsequent ideal action.

In contrast to the example of chess, where the present state of the board is sufficient to identify the next optimal move for a player, in this reinforcement learning scenario, the state of the agent is only partially observable, meaning that there is no explicitly stated state of the agent. Because of this, the agent's current state must be deduced from its observations or from knowledge of the past. In order to determine the present status of the network and Deep Learning, the historical pricing series of Bitcoin are accepted as input.

1.2.1 Input

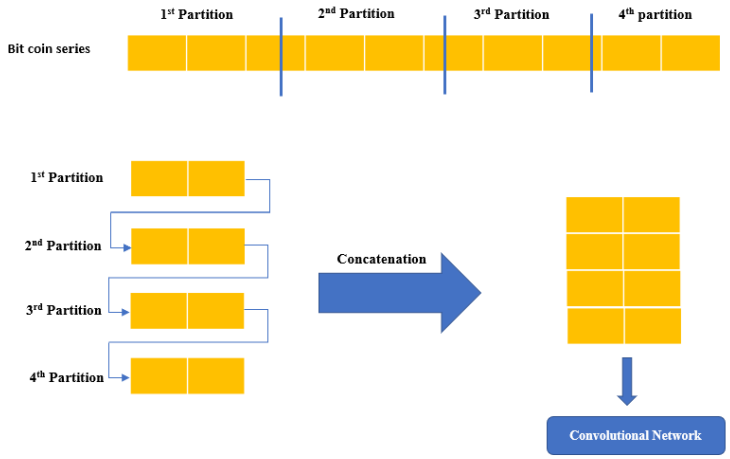
The number of past price entries taken from the current timestamp to produce the representation of the state has a fixed time horizon (depending on the frequency and the number of entries in the time series). In other words, the amount of prior Bitcoin prices needed to determine an agent's state at any timestamp is set, and this time series is fed into Deep Sense to produce an agent's state representation.

In addition to previous prices, Deep Sense also receives data from popular technical indicators including the Simple Moving Average (15 to 60 periods), Relative Strength Index, and Average True Range.

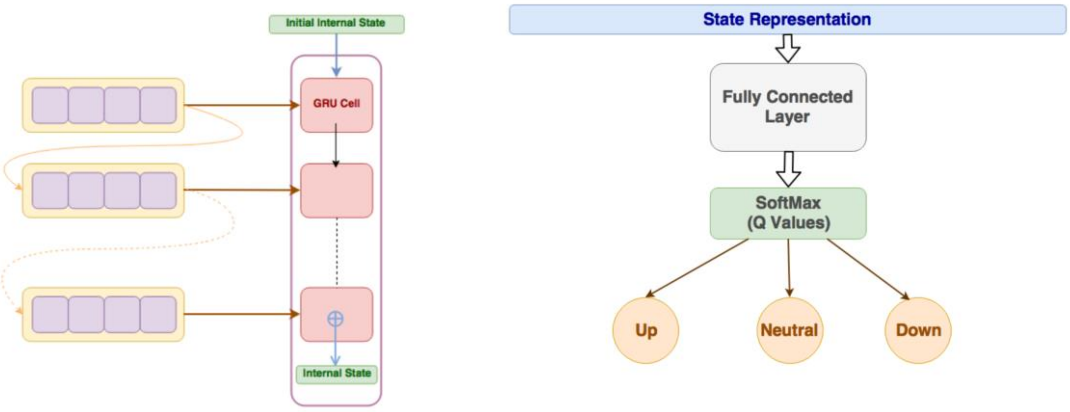
1.3 Deep Sense

The input time series are divided using the timestamps by the Deep Sense architecture.

1. Creates a high level representation of each partition that includes all intra partition dependencies by processing each partition using a multi-layered convolutional network.



2. Combines the high level representations of each partition acquired from the convolutional network into a single vector using a recurrent neural network (stacked GRU in this example) (the final internal state of the Recurrent Network) The agent's current state is represented by the Recurrent Network's ultimate internal state.



We are currently working on the bitcoin series partitioning and generating convolutional network.

2 Challenges

During the implementation of a reinforcement learning project for trading, several challenges may arise:

1. Quality of Data: Data quality and quantity can have a significant impact on the performance of reinforcement learning algorithms. A suboptimal trading strategy can result from incomplete, noisy, or biased data.
2. Non-stationary environments: Financial markets are dynamic and often exhibit non-stationary behavior, which can make it challenging for reinforcement learning algorithms to converge to an optimal solution.
3. Reward Design: The reward function is crucial to the success of a reinforcement learning algorithm in trading. To design a reward function that incentivizes the agent to learn profitable trading strategies while accounting for risk management is not an easy task.
4. Overfitting: Reinforcement learning algorithms are prone to overfitting, which can lead to poor generalization to unseen market conditions.
5. Computational Resources: Reinforcement learning algorithms can be computationally expensive, particularly when using deep neural networks. Training a deep reinforcement learning agent can require significant computational resources, which can be costly.
6. Regulation: The use of reinforcement learning in trading may be subject to regulatory constraints, particularly if the algorithm makes decisions that affect market behavior.
7. Ethical Considerations: The use of reinforcement learning in trading raises ethical considerations, particularly if the algorithm is making decisions that could impact society as a whole. In order to ensure that these algorithms are aligned with ethical principles, it is important to take into account the potential consequences of using them in the trading process.

3 Next Steps

The following could be the next steps for this reinforcement learning project after the existing issues have been resolved:

The incentive for the agent to do a specific action is determined by the reward function, which is defined below. In this scenario, the reward function must be created in a way that motivates the agent to operate in a way that maximizes the value of the portfolio.

Implementing the reward function: The reinforcement learning algorithm must include the reward function once it has been defined. This entails altering the state-action pair Q-values in accordance with the rewards the agent has earned. Although frequent intermediate rewards are frequently noisy and tend to destabilize the trading process, non-zero intermediate rewards enable the agent to converge to a trading strategy in fewer iterations.

The reward supplied to the agent at any timestamp represents the portfolio's true value. Equation serves as its definition.

$$R_n = \big(l(t_n) - s(t_n)\big) \times \big(p(t_n) - p(t_{n-1})\big)$$

Where,

- $l(t_i)$ be the amount of long currency,
- $s(t_i)$ be the amount of short currency and
- $p(t_i)$ be the price of the currency at time instant t_i.

Adjusting the hyperparameters: To attain the best performance, reinforcement learning algorithms' various hyperparameters must be tweaked. The learning rate, discount factor, exploration rate, and batch size are a few of them. The algorithm's performance can be greatly enhanced by tweaking these hyperparameters.

After the reinforcement learning algorithm has been trained, it must be tested and assessed on a different dataset to determine how well it performs. Measuring criteria for this include maximum drawdown, Sharpe ratio, and portfolio returns.

After being tested and assessed, the reinforcement learning algorithm can be put into use in a live trading environment.

However, it is crucial to properly test the algorithm in a simulated trading environment before deployment to make sure it functions as planned and does not result in any unforeseen losses.