Multi-Agent-Reinforcement-Learning

Value Based and Policy Based reinforcement learning

In value based reinforcement learning, the exploration process tries to determine a value function that represents the expected cumulative reward of being in a state. Value based reinforcement learning doesn't have an explicit policy. The policy is implicitly defined by the value function.

> The value function can be a state value function or an state-action value function. The difference between the two is the input to the function. The state value function takes a state as input and returns the expected cumulative reward of being in that state. The state-action value function takes a state and an action as input and returns the expected cumulative reward of being in that state and taking that action. They are of the form
>
> The state value function is of the form $V(s)$. The state-action value function is of the form $Q(s, a)$. Where s is the state and a is the action.

Policy based reinforcement learning consists of an explicit policy that the agent follows. The policy is a mapping from states to actions. The policy can be deterministic or stochastic. In the deterministic case, the policy is a function that takes a state as input and returns an action. In the stochastic case, the policy is a probability distribution over actions given a state.

Bellman Equation

Bellman equation is used to update the value function. The Bellman equation is a recursive equation that relates the value of a state to the value of its successor states. Depending on the type of value function and the other aspects such as the type of the policy, the Bellman equation can take different forms.

Q - Learning

Q-Learning is a value based reinforcement learning algorithm. That means instead of learning a policy, it learns a value function. The value function is called the Q-function. The Q-function is a state-action value function. The Q-function is updated using the Bellman equation. Q-Learning is an off policy algorithm. That means the agent can learn from the experiences of another policy. The policy that the agent follows during exploration is called the behavior policy. The policy that the agent is trying to learn is called the target policy.
The Q-function is given by

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Where G_t is the expected cumulative reward from time t onwards following policy π, S_t is the state at time t and A_t is the action at time t.

The Q-function is updated using the Bellman equation. The Bellman equation for the Q-function is given by

$$Q(s_t, a_t) = R_{t+1} + \gamma \max_a Q(s_{t+1}, a)$$

Where s_t is the state at time t, a_t is the action at time t, r_{t+1} is the reward at time $t + 1$.

Deep Q - Learning

Deep Q-Learning is a value based reinforcement learning algorithm that uses a neural network to approximate the Q-function. This is done to handle the large number of state-action pairs in the state space.

Episodic and Non-Episodic Tasks

In episodic tasks, the agent interacts with the environment for a finite number of time steps representing an episode. The episode ends when the agent reaches a terminal state. In non-episodic tasks, there is no explicit end to the interaction.

> In the case of financial trading, if we treat it as a episodic task, it's possible that the agent learns the optimal policy for a particular time period (training data) and doesn't perform well for out of period data.

Approach taken in this project

Assumptions

1. The paper assumes that there is no transaction cost.
2. Every order that is performed uses the entire capital available to the agent.
3. The market impact of the orders is considered to be negligible.

The paper frames the trading problem as a MDP with a (S, A, P, R, γ) where S is the the set of available states, A is the set of actions, P is the transition probability matrix, R is the reward function and γ is the discount factor.

1. The state at time t is given by the collection of the last n OHLCV bars and the decisions made by the agents in the higher time frames.
2. The action at time t is given by the set of actions $\{0, 1, -1\}$ where 0 represents holding, 1 represents opening a long position and -1 represents opening a short position, 0 representing closing the previous position. Each position is held for a time t_w.

   > The paper defines t_w as the "freezing period" of the position. This is the time period for which the position is held. This is done to better handle noise. This is also similar to real world trading where positions are held for a certain time period.

3. The transition probability matrix, since we are using DQN which is a model free algorithm, is not explicitly defined.
4. The reward function is given by

$$R_t = \left(\frac{P_{t+t_w} - P_t}{P_t} \right) \cdot A_t$$

   Where P_t is the price at time t, A_t is the action at time t and t_w is the freezing period of the position.

The paper uses grid search to find the optimal hyperparameters for the DQN algorithm.

The paper uses a multi-agent reinforcement learning approach to solve the problem. Each agent uses DQN to approach the trading problem. The primary

difference between the agents is the timeframes they use to make decisions. Further, each agent on a lower time frame has access to the decisions made by the agents on higher time frames, this is done to ensure a form of information flow from higher time frames to lower time frames. According to the paper, the approach is inspired by the fractal nature of financial markets.

> For example, say one agent uses a 5 minute timeframe to make decisions, that means it uses data points that are each separated by 5 minutes to make decisions. Another agent might use a 1 hour timeframe to make decisions.