A reinforcement learning based approach to optimizing vAMMs (WIP)

mystericalfox

December 2021

Abstract

We aim to model the vAMM (or an entirely configurable vAMM, also known as a dAMM) as a reinforcement learning/Q-learning problem, from which we can determine an optimal policy (ideally for parameterizing the peg multiplier/liquidity depth based on trading activity).

1 Previous Applications of Q-learning in Finance

The idea of using deep learning techniques in traditional finance isn't necessarily a new one; for instance, Mani et al. [2019] introduced a framework for automated market-making using risk-sensitive reinforcement learning. In particular, they established an agent (market maker)-environment (trading activity) model by defining it as a finite Markov decision process in discrete time with a linear reward function, and solving it using temporal difference learning algorithms. Moreover, Zhang et al. [2019] have proposed using deep reinforcement learning to design trading strategies for continuous futures contracts, considering both discrete and continuous action spaces.

2 Proposed Model

2.1 Typical Setup

In a typical portfolio-maximization problem, we might expect an agent to choose the sequence of actions that will maximize the expected utility of wealth at the end of finite time horizon T:

$$E[U(W_T)] = E[U(W_0 + \sum_{t=1}^{T} \Delta W_t)]$$

We aim to take a similar approach in our model by defining the concept of expected "wealth" (i.e. immediate reward) in the vAMM setting.

2.2 Notation

We aim to define the vAMM as a finite stationary Markov Decision Process (MDP), described by the tuple $M = \langle S, A, R, P \rangle$, where:

- S is a finite collection of states; in this case, the long-short imbalance at time t.

$$S_t = IMB_t = \text{long open interest}_t - \text{short open interest}_t$$

- A is a finite collection of actions; in this case, the agent's choice to long or short at time t.

$$A_t = \begin{cases} +1 & \text{for a long position} \\ -1 & \text{for a short position} \end{cases}$$

- R is the set of possible values of immediate rewards; in this case we consider "reward" to mean the fee pool (in Drift Protocol, this is 0.1% of trading volume) capped by the traunch limit (in Drift Protocol, this is 50%) at time t.

$$R_t = \text{trading volume}_t \cdot 0.001 \cdot 0.5$$

- P describes the transition probabilities i.e. $P(s', r \mid s, a) = P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$ for $s, s' \in S, a \in A, r \in R$. We expect to approximate this numerically from Drift Protocol's trading data.

Furthermore, we assume that:

- Informed traders enter the market with probability λ_i, take a long position if $mark < oracle$ and a short position if $mark > oracle$.
- Uninformed traders enter the market with probability λ_u and have an even chance of taking a long or short position.

We assume that these are mutually exclusive events such that $\lambda_i + \lambda_u = 1$.

The agent-environment dynamics is as follows: at time t, the market maker observes a state S_t and takes an action A_t according to a Markovian policy, from which a numerical reward R_{t+1} is provided at the next time step, leading to the next state S_{t+1}.

At any time t, the goal is to maximize the expected return, denoted as:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1}(\lambda_i R_k + \lambda_u R_k)$$

where $\gamma \in (0, 1)$ is a discount factor representing the agent's uncertainty about the time horizon remaining on the contract. We weight the rewards accordingly by the probability of observing a group of informed or uninformed traders, as we expect the immediate rewards to differ greatly based on strategic trading activity. Assuming that the utility of final wealth is linear over time, we can see that optimizing $E[G]$ is also equivalent to optimizing our expected wealth.

2.3 Optimal Policy

Define the state-action value function $Q_\pi : S \times A \to \mathbb{R}$ for some fixed policy π as follows:

$$\begin{aligned} Q_\pi(s, a) &= E_\pi[\sum_{k=t+1}^{T} \gamma^{k-t-1}(\lambda_i R_k + \lambda_u R_k) \mid S_t = s, A_t = a] \\ &= E_\pi[R_{t+1} + \gamma \sum_{a'} Q_\pi(S_{t+1}, a')\pi(a' \mid S_{t+1}) \mid S_t = s, A_t = a] \end{aligned}$$

Thus the optimal policy is determined by:

$$\begin{aligned} Q_*(s, a) &= \max_\pi Q_\pi(s, a) \\ &= E[R_{t+1} + \gamma \max_{a'} Q_*(S_{t+1}, a') \mid S_t = s, A_t = a] \\ \implies \pi_* &= \text{argmax}_a Q_*(s, a) \end{aligned}$$

3 References

1. Perpetual Protocol vAMM guide: `https://medium.com/perpetual-protocol/a-deep-dive-into-our-virtual-amm`
2. Paradigm Intro to AMMs: `https://www.paradigm.xyz/2021/04/understanding-automated-market-makers-part-`
3. Paradigm Report on Uniswap v3: `https://www.paradigm.xyz/2021/06/uniswap-v3-the-universal-amm/`
4. Applications of Reinforcement Learning in Automated Market Making: `http://www.agent-games-2019.preflib.org/wp-content/uploads/2019/05/GAIW2019_paper_5.pdf`
5. Deep Reinforcement Learning for Trading: `https://arxiv.org/pdf/1911.10107.pdf`