Stanford CME 241 (Winter 2023) - Assignment 6

Due: 2/20 @ 11:59pm. Solve all 3 questions.

1. We consider a special case of the Optimal Market-Making problem we covered in class (Avellandea-Stoikov formulation) where the market-maker has a cash amount of $W \in \mathbb{R}$ at time 0 and an inventory of shares equal to $I \in \mathbb{Z}$ (note: this could be positive or negative), but is not going to be market-making until time T. The market maker's *Value Function* at time t (with $0 \leq t \leq T$) is given by the Expected Utility at time T (conditional on the time t and the OB Mid Price S_t at time t):

$$V(t, S_t, W, I) = \mathbb{E}[-e^{-\gamma \cdot (W + I \cdot S_T)} | (t, S_t)]$$

Assume the same process for the OB Mid Price as we had covered in class:

$$dS_t = \sigma \cdot dz_t$$

for some fixed $\sigma \in \mathbb{R}^+$. This means:

$$S_{t_2} \sim \mathcal{N}(S_{t_1}, \sigma^2 \cdot (t_2 - t_1))$$

for all $0 \leq t_1 \leq t_2$.

- Under this process for S_t, evaluate the conditional expectation $\mathbb{E}[-e^{-\gamma \cdot (W + I \cdot S_T)} | (t, S_t)]$ so you have a simple expression for $V(t, S_t, W, I)$.
- Using the above expression for $V(t, S_t, W, I)$, calculate the Indifference Bid Price $Q^{(b)}(t, S_t, I)$ and the Indifference Ask Price $Q^{(a)}(t, S_t, I)$, which have the same definitions as we had covered in class, as follows:

$$V(t, S_t, W - Q^{(b)}(t, S_t, I), I + 1) = V(t, S_t, W, I)$$
$$V(t, S_t, W + Q^{(a)}(t, S_t, I), I - 1) = V(t, S_t, W, I)$$

2. We'd like to test the performance of the Optimal Policy we derived in class for the Optimal Market-Making problem. In particular, we want to compare this Optimal Policy against a policy (call it "Naive Policy") that is always symmetric around the OB Mid Price (rather than around the Indifference Price) with a constant Bid-Ask Spread equal to the average Bid-Ask Spread of the Optimal Policy.

   We will do the comparison by generating a large number of simulation traces. Each simulation trace consists of $\frac{T}{\Delta t}$ time steps. The time step Δt needs to be small enough so that the probability of multiple orders transacting with the market-maker is small but needs to be large enough so that there are indeed orders that transact with the market-maker. In their paper, Avallaneda-Stoikov point out that $\Delta t = 0.005$ worked well in their simulation experiments.

   Here are the details on how to perform a time step in each simulation trace for the Optimal Policy:

   - At each time t, we observe the *State*, and calculate the Optimal Action $(P_t^{(b)^*}, P_t^{(a)^*})$.

- With probability $c \cdot e^{-k \cdot \delta_t^{(a)^*}} \cdot \Delta t$, the inventory variable is decremented by 1 and the trading PnL is increased by $P_t^{(a)^*}$.
- With probability $c \cdot e^{-k \cdot \delta_t^{(b)^*}} \cdot \Delta t$, the inventory variable is incremented by 1 and the trading PnL is decreased by $P_t^{(b)^*}$.
- The OB Mid Price is incremented or decremented randomly (each with probability 0.5) by $\sigma \cdot \sqrt{\Delta t}$.
- These updates to the inventory variable, to the trading PnL and to the OB Mid Price give us the *State* for the next time $t + \Delta t$.

We run a large number (say 10,000) such simulation traces. We calculate the average Bid-Ask Spread across all time steps across all simulation traces. Then we set the Bid-Ask Spread for the "naive policy" to be this average Bid-Ask Spread (constant Bid-Ask Spread). Repeat the same large number of simulations for the "naive policy".

Track the relevant metrics for each simulation trace on both the policies - the trading PnL, the Inventory, the OB Mid Price, the Bid Price, the Ask Price, the number of hits and lifts etc. Plot graphs for these metrics. You can view the metrics on a single simulation trace or you can view the average metrics at a fixed time (in particular for terminal time T). Demonstrate empirically that the Optimal Policy does indeed perform better than the "naive policy".

Avallaneda-Stoikov used the following parameters in their simulation:

$S_0 = 100, T = 1, \Delta t = 0.005, \gamma = 0.1, \sigma = 2, I_0 = 0, k = 1.5, c = 140.$

3. Although the provided code contains a number of utilties for Monte-Carlo prediction, you may consider writing re-writing these functions to improve your understanding.

   a. We have written the function mc_prediction in rl/monte_carlo.py as an implementation of Monte-Carlo Prediction with function approximation. You have also learnt that since Tabular MC Prediction is a special case of MC Prediction with Function Approximation and so, writing a separate function for Tabular MC Prediction is not necessary. But for a learning experience, it's a great idea to write a function for Tabular MC Prediction **from scratch**. Think about what the input and output types must be. Be sure to reduce the learning rate appropriately as a function of number of updates (or as a function of number of episodes).

   b. We have written the function td_prediction in rl/td.py as an implementation of Temporal-Difference Prediction with function approximation. You have also learnt that since Tabular TD Prediction is a special case of TD Prediction with Function Approximation and so, writing a separate function for Tabular TD Prediction is not necessary. But for a learning experience, it's a great idea to write a function for Tabular TD Prediction **from scratch**. Think about what the input and output types must be. Be sure to reduce the learning rate appropriately as a function of number of updates.

   c. Test your above implementations of Tabular MC Prediction and Tabular TD Prediction on SimpleInventoryMRPFinite (from rl/chapter2/simple_inventory_mrp.py) by ensuring that your Value Function output matches that produced by the function approximation versions of MC Prediction and TD Prediction.