Stanford CME 241 (Winter 2022) - Assignment 13

Assignments:
Do 2 of {1,2,3}

1. Implement Tabular Monte-Carlo Control algorithm in Python with GLIE implemented as $\epsilon = \frac{1}{k}$ for episode number k and initial state of each episode sampled uniformly from the state space. Test your implementation against the Optimal Value Function/Optimal Policy obtained by DP on SimpleInventoryMDPCap in rl/chapter3/simple_inventory_mdp_cap.py. Next, generalize your implementation to MC Control with Function Approximation (your interface should be similar to the interface we had designed for MC Prediction with Function Approximation). Test your implementation against the Optimal Value Function/Optimal Policy obtained by ADP on AssetAllocDiscrete in rl/chapter7/asset_alloc_discrete.py.

2. Implement Tabular SARSA algorithm in Python with GLIE and a parameterized trajectory of decreasing step sizes. Test your implementation against the Optimal Value Function/Optimal Policy obtained by DP on SimpleInventoryMDPCap in rl/chapter3/simple_inventory_mdp_cap.py. Next, generalize your implementation to SARSA with Function Approximation (your interface should be similar to the interface we had designed for TD Prediction with Function Approximation). Test your implementation against the Optimal Value Function/Optimal Policy obtained by ADP on AssetAllocDiscrete in rl/chapter7/asset_alloc_discrete.py.

3. Implement Tabular Q-Learning algorithm in Python with infinite exploration of all (state, action) pairs and with a parameterized trajectory of decreasing step sizes. Test your implementation against the Optimal Value Function/Optimal Policy obtained by DP on SimpleInventoryMDPCap in rl/chapter3/simple_inventory_mdp_cap.py. Next, generalize your implementation to Q-Learning with Function Approximation (your interface should be similar to the interface we had designed for TD Prediction with Function Approximation). Test your implementation against the Optimal Value Function/Optimal Policy obtained by ADP on AssetAllocDiscrete in rl/chapter7/asset_alloc_discrete.py

4. **Optional**: Assume you are the owner of a bank where customers come in randomly everyday to make cash deposits and to withdraw cash from their accounts. At the end of each day, you can borrow (from another bank, without transaction costs) any cash amount $y > 0$ at a constant daily interest rate R, meaning you will need to pay back a cash amount of $y(1 + R)$ at the end of the next day. Also, at the end of each day, you can invest a portion of your bank's cash in a risky (high return, high risk) asset. Assume you can change the amount of your investment in the risky asset each day, with no transaction costs (this is your mechanism to turn any amount of cash into risky investment or vice-versa). The key point here is that once you make a decision to invest a portion of your cash in the risky asset at the end of a day, you will not have access to this invested amount as cash that otherwise could have been made available to customers who come in the next day for withdrawals. More importantly, if the cash amount c in your bank at the start of a day is less than C, the banking regulator will make you pay a penalty of $K \cdot \cot(\frac{\pi \cdot c}{2C})$ (for a given constant $C > 0$ and a given constant $K > 0$).

   For convenience, we make the following assumptions:

   - Assume that the borrowing and investing is constrained so that we end the day (after borrowing and investing) with positive cash ($c > 0$) and that any amount of regulator penalty can be immediately paid (meaning $c \geq K \cdot \cot(\frac{\pi \cdot c}{2C})$ when $c \leq C$).

- Assume that the deposit rate customers earn is so small that it can be ignored.
- Assume for convenience that the first half of the day is reserved for only depositing money and the second half of the day is reserved for only withdrawal requests.
- Assume that if you do not have sufficient cash to fulfill a customer withdrawal request, you ask the customer to make the withdrawal request again the next day.
- Assume all quantities are continuous variables.

Your first task is to model an MDP so you can run the bank in the most optimal manner, i.e., maximizing the Expected Utility of assets less liabilities at the end of a T-day horizon, conditional on any current situation of assets and liabilities. Specify the states, actions, transitions, rewards with precise mathematical notation (make sure you do the financial accounting from one day to the next precisely).

In a practical setting, we do not know the exact probability distributions of the customer deposits and withdrawals. Neither do we know the exact stochastic process of the risky asset. But assume we have access to a large set of historical data detailing daily customer deposits and withdrawal requests, as well as daily historical market valuations of the risky asset. Assume we also have data on new customers as well as leaving customers (sometimes due to their withdrawal requests not being satisfied promptly). Solve this problem with Reinforcement Learning by constructing and then using the historical data described above.