Assignment 12

Lilian Kourti
CME 241

Problem 1

GLIE Tabular Monte-Carlo Control is implemented in prob13_1.py. It was tested against the Optimal Value Function/Optimal Policy obtained by Value Iteration on AssetAllocDiscrete in rl/chapter7/asset_alloc_discrete.py. Value Iteration gives the following output:

```
MDP Value Iteration Optimal Value Function and Optimal Policy
--------------
{InventoryState(on_hand=0, on_order=1): -27.66095021630144,
 InventoryState(on_hand=0, on_order=0): -34.89484576629397,
 InventoryState(on_hand=0, on_order=2): -27.991890076067463,
 InventoryState(on_hand=1, on_order=0): -28.660950216301437,
 InventoryState(on_hand=1, on_order=1): -28.991890076067467,
 InventoryState(on_hand=2, on_order=0): -29.991890076067463}
For State InventoryState(on_hand=0, on_order=0):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=0, on_order=1):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=0, on_order=2):
  Do Action 0 with Probability 1.000
For State InventoryState(on_hand=1, on_order=0):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=1, on_order=1):
  Do Action 0 with Probability 1.000
For State InventoryState(on_hand=2, on_order=0):
  Do Action 0 with Probability 1.000
```

The GLIE Tabular Monte-Carlo Control gives the following output:

```
InventoryState(on_hand=0, on_order=0): {0: -47.32024594968155, 1: -38.0864767370392, 2: -35.781108743668554}
InventoryState(on_hand=0, on_order=1): {0: -34.376514944471914, 1: -28.225973171869594}
InventoryState(on_hand=0, on_order=2): {0: -28.720648821820966}
InventoryState(on_hand=1, on_order=0): {0: -36.73568500060274, 1: -29.042532924906403}
InventoryState(on_hand=1, on_order=1): {0: -29.494616268337666}
InventoryState(on_hand=2, on_order=0): {0: -30.409674884357532}
```

As it can be observed it converges towards the right direction.

Problem 2

The implementation for SARSA can be found in prob13_2.py. It was tested against the Optimal Value Function/Optimal Policy obtained by Value Iteration on AssetAllocDiscrete in rl/chapter7/asset_alloc_discrete.py. Value Iteration gives the following output:

```
MDP Value Iteration Optimal Value Function and Optimal Policy
--------------
{InventoryState(on_hand=0, on_order=1): -27.66095021630144,
 InventoryState(on_hand=0, on_order=0): -34.89484576629397,
 InventoryState(on_hand=0, on_order=2): -27.991890076067463,
 InventoryState(on_hand=1, on_order=0): -28.660950216301437,
 InventoryState(on_hand=1, on_order=1): -28.991890076067467,
 InventoryState(on_hand=2, on_order=0): -29.991890076067463}
For State InventoryState(on_hand=0, on_order=0):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=0, on_order=1):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=0, on_order=2):
  Do Action 0 with Probability 1.000
For State InventoryState(on_hand=1, on_order=0):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=1, on_order=1):
  Do Action 0 with Probability 1.000
For State InventoryState(on_hand=2, on_order=0):
  Do Action 0 with Probability 1.000
```

The SARSA method gives the following output:

```
InventoryState(on_hand=0, on_order=0): {0: -41.44879519571637, 1: -34.89599360274678, 2: -35.276058213482514}
InventoryState(on_hand=0, on_order=1): {0: -31.431760584405637, 1: -27.61491321906224}
InventoryState(on_hand=0, on_order=2): {0: -27.95887165458977}
InventoryState(on_hand=1, on_order=0): {0: -33.394070624258894, 1: -28.638828069193014}
InventoryState(on_hand=1, on_order=1): {0: -28.954479542983897}
InventoryState(on_hand=2, on_order=0): {0: -30.045814855049922}
```

As it can be observed it converges to the true value function.

Problem 3

The implementation for Q-Learning can be found in prob13_3.py. It was tested against the Optimal Value Function/Optimal Policy obtained by Value Iteration on AssetAllocDiscrete in rl/chapter7/asset_alloc_discrete.py. Value Iteration gives the following output:

```
MDP Value Iteration Optimal Value Function and Optimal Policy
--------------
{InventoryState(on_hand=0, on_order=1): -27.66095021630144,
 InventoryState(on_hand=0, on_order=0): -34.89484576629397,
 InventoryState(on_hand=0, on_order=2): -27.991890076067463,
 InventoryState(on_hand=1, on_order=0): -28.660950216301437,
 InventoryState(on_hand=1, on_order=1): -28.991890076067467,
 InventoryState(on_hand=2, on_order=0): -29.991890076067463}
For State InventoryState(on_hand=0, on_order=0):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=0, on_order=1):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=0, on_order=2):
  Do Action 0 with Probability 1.000
For State InventoryState(on_hand=1, on_order=0):
  Do Action 1 with Probability 1.000
For State InventoryState(on_hand=1, on_order=1):
  Do Action 0 with Probability 1.000
For State InventoryState(on_hand=2, on_order=0):
  Do Action 0 with Probability 1.000
```

The Q-Learning method gives the following output:

```
InventoryState(on_hand=0, on_order=0): {0: -39.02832248547219, 1: -34.96846215171993, 2: -35.10442804950834}
InventoryState(on_hand=0, on_order=1): {0: -30.25372856118806, 1: -27.62391427819714}
InventoryState(on_hand=0, on_order=2): {0: -27.975307408694704}
InventoryState(on_hand=1, on_order=0): {0: -31.80432480149277, 1: -28.644078682736552}
InventoryState(on_hand=1, on_order=1): {0: -28.941112138958943}
InventoryState(on_hand=2, on_order=0): {0: -29.93858006061513}
```

As it can be observed it converges to the true value function.