Technical University of Crete
School of Electrical and Computer Engineering
Course: **Reinforcement Learning and Dynamic Optimization**

Assignment 3 - Phase 1

Angelopoulos Dimitris - 2020030038
Michalochristas Konstantinos - 2020030111

In this exercise we implement a Markov Decision Process environment to model a stock market of N stocks. We assume full environment knowledge.

Task 1 First and foremost we need to model our MDP environment. This considered, we have to define a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. In the sequel we break down each component of $\mathcal{M}$.

1. State Space $\mathcal{S}$: To capture all the information of the stock market environment, we need to consider the following: (i) the Markov state of every stock. For every combination of each stock's Markov state we have to keep track of (ii) the stock we are currently holding. That said, for N stocks, we get 2^N Markov states that we must track for every one of them. This yields $|\mathcal{S}| = N2^N$ number of states, $S_0 = \{0, H, H, \ldots, H\}$, $S_1 = \{0, H, H, \ldots, L\}$, $\ldots$, $S_{N2^N-1} = \{N-1, L, L, \ldots, L\}$.

2. Action Space $\mathcal{A}$: For this set we just need to keep the action of which stock we want to invest in. For example, if we want to invest in stock 5, then $a = 5$. Thus $\mathcal{A} = \{0, 1, \ldots, N-1\}$.

3. Transition Probability Function $\mathcal{P}$: This function takes as inputs the current state s and the action we take in this time instant, and returns the transition probability to some next state s'. For example, for $N = 2$ stocks, if we currently are at state $s = \{0, H, H\}$ and choose to switch the stock we hold ($a = 1$) then all the possible next states are a subset of $\mathcal{S}$. This subset is $\mathcal{S}' = \{\{1, H, H\}, \{1, H, L\}, \{1, L, H\}, \{1, L, L\}\}$ (the rest of the states have zero probability to transition due to the action). For $s' = \{1, L, H\}$ we have $\mathcal{P}(s'|a, s) = p^0_{HL} p^1_{HH}$. The sum of the transition probabilities to all possible s' states is equal to one.

4. Transition Reward Function $\mathcal{R}$: This function has the same arguments as the previous one. Essentially, here we need knowledge of the Markov state of the stock we want to start investing to. For example, if the stock we switch to is at Markov state H then $\mathcal{R}(s, a, s') = r_a^H - c$, where $c \in [0, 1]$ is the transaction fee. If the stock we switch to is the stock we are currently holding then no transaction fee is applied and $\mathcal{R}(s, a, s') = r_a^H$

5. Terminal States: There are none.

Task 2,3 In these tasks we wrote code that generates the environment stated in the previous task and our agent. Our agent has full knowledge of the environment and is essentially the Policy Iteration algorithm. The PI algorithm consists of two other algorithms. The first one is the policy evaluation algorithm which takes as input a given policy $\pi(s)$ and estimates the value function $V(s)$ using dynamic programming. The second algorithm is policy improvement, which takes as input the evaluated $V(s)$ and returns a new improved policy. In the PI algorithm when the old policy (input of policy evaluation) is the same with the improved one, then this policy is the optimal and the algorithm is terminated.

Question 1 We assume that $N = 2$ and $r_0^H = 2r_1^H = 0.08$. We will find the parameters for which the optimal policy always suggests that we do not switch to another stock. The case where this holds primarily depends on the tuning of the transaction fee c. As we increase c from 0.01 to 0.04 we can observe that our policy becomes more conservative. When c approaches the value of r_1^H, the evaluated optimal policy is always to stay. That means that for $c \in [0.04, 1]$ and for the parameters $r_0^L = -0.04$, $r_1^L = 0.01$ and $p_{HH}^0 = 0.7$, $p_{HL}^0 = 0.3$, $p_{LH}^0 = 0.3$, $p_{LL}^0 = 0.7$, $p_{HH}^1 = 0.8$, $p_{HL}^1 = 0.2$, $p_{LH}^1 = 0.4$, $p_{LL}^1 = 0.6$, π^* suggests to not switch stock.

Question 2 For the parameters of the previous question for $N = 2$ and $\gamma = 0.9$, our agent yields the following policy:

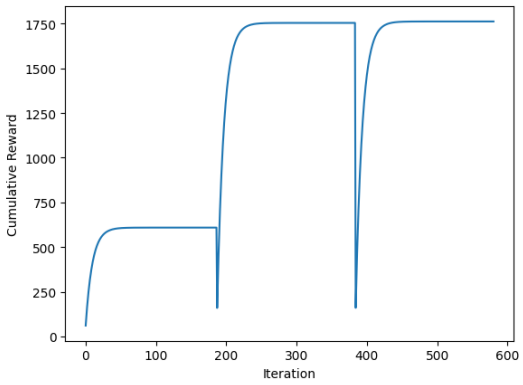
$$\pi^*(s) = \{S_0 : 0, S_1 : 0, S_2 : 1, S_3 : 1, S_4 : 0, S_5 : 0, S_6 : 1, S_7 : 1\}$$

To confirm our results we will pick some state s and we will compute the expected reward given a certain action a. Suppose $s = \{0, H, H\}$.

- For $a = 0$, the expected reward is $\mathbb{E}[\mathcal{R}] = p^0_{HH} r^H_0 + p^0_{HL} r^L_0 = 0.044$
- For $a = 1$, the expected reward is $\mathbb{E}[\mathcal{R}] = p^1_{HH} r^H_1 + p^1_{HL} r^L_1 - c = 0.024$

We can see that for the action $a = 0$ we get higher expected reward, meaning that the best action is to keep holding stock 0, verifying the action above. Likewise for the rest of the states we can verify that our agent indeed finds the optimal policy.

Question 3 We will generate $r^H_i, r^L_i \sim \mathcal{U}[-0.02, 0.1]$ and probabilities p^i_{HL}, p^i_{LH} equal to 0.1 for half the stocks and 0.5 for the other half, $\forall i = 0, \ldots, N-1$. We run the PI algorithm for $N = 8$ stocks an get the following graph that represents the cumulative reward that is evaluated through the value function in the policy evaluation algorithm.



In the figure above, each time the accumulated reward is zero, a new PI iteration starts. Essentially the graph shows us how every improved policy yields a reward. For the last policy iteration we get the best cumulative reward and thus the optimal policy.

The Colab code for the assignment can be found in the following link.