CS520 - INTRO TO ARTIF INTEL

Final: Question2-Markov Decision Processes

Name:
Xinyu Lyu(xl422)
182007396

You are a robot. One of your parts is a machine that can be in any one of ten possible states:

New, Used1, Used2, Used3, Used4, Used5, Used6, Used7, Used8, Dead

- **In states New and Used1,...,Used8, you can USE the machine. In states Used1,...,Used8, Dead, you can REPLACE the machine.**
- **In any state, the action REPLACE sends the machine to state New with probability 1, at cost 250.**
- **In state New, USE gives a reward of 100, and transitions to state Used1 with probability 1. For i = 1, . . . , 7, in state Usedi, taking action USE yields a reward of 100 − 10 ∗ i, and with probability 0.1 ∗ i the machine transitions to state Usedi+1, otherwise stays in state Usedi. In state Used8, action USE yields a reward of 20, and transitions to state Dead with probability 0.8, otherwise stays in Used8 with probability 0.2.**
- **The only action available in state Dead is REPLACE.**
- **At every time step, all future rewards (and costs) are discounted at a factor of β = 0.9.**

a) For each of the 10 states, what is the optimal utility (long term expected discounted value) available in that state (i.e., U∗(state))?

Algorithm VALUE-ITERATION

Function Value-Iteration (***mdp***, ε) **returns** a utility function

Inputs : ***mdp***, an MDP with states ***S***, actions ***A(s)***, transition model ***P(*** s' **|s ,a)**, rewards ***R(s)***, discount γ

ε, the maximum error allowed in the utility of any state ***S***

Local Variables: U , U' vectors of utilities for states in ***S***, initially

δ, the maximum change in the utility of any state in an iteration

Repeat:

$U \leftarrow U'$; $\delta \leftarrow 0$

For each state **s in *S* do:**

$U'[s] \leftarrow max_{a \in A(s)} \sum_{s',r} P(s', r \mid s, a)[R(s) + \gamma U(s')]$

If $|U'[s] - U(s')| > \delta$ **,then** $\delta \leftarrow |U'[s] - U(s')|$

Until $\delta < \varepsilon(1-\gamma)/\gamma$

Return *U*

We implement the **Algorithm** VALUE-ITERATION shown above. States ***S*** is the possible states as New, Used1, Used2, Used3, Used4, Used5, Used6, Used7, Used8, Dead. actions A(s) are replaced and use operations. transition model P(s' | s, a) is the probability of changing from State s' to state S with the action. discount γ is the β. And U' is the update utility in each iteration. In the loop, we update the utility U with U' and δ becomes smaller and smaller. The iteration doesn't stop until $\delta < \varepsilon(1-\gamma)/\gamma$. Then we get the optimal utility for

each State S together with the corresponding optimal actions A(s). And we set ε as 0.00001.

After running the program, we get the result of the optimal utility for each $U_*(State)$.

$U_*(New)$ = 800.5316006249429

$U_*(Used_1)$ = 778.368446166737

$U_*(Used_2)$ = 643.2222855200046

$U_*(Used_3)$ = 556.1235603928982

$U_*(Used_4)$ = 502.8359935944176

$U_*(Used_5)$ = 475.8459943848679

$U_*(Used_6)$ = 470.4784396373283

$U_*(Used_7)$ = 470.4784396373283

$U_*(Used_8)$ = 470.4784396373283

$U_*(Dead)$ = 470.4784396373283

b) What is the optimal policy that gives you this optimal utility - i.e., in each state, what is the best action to take in that state?

Algorithm POLICY-ITERATION

Function Policy-Iteration (***mdp***) **returns** a policy

Inputs : ***mdp***, an MDP with states ***S***, actions ***A(s)***, transition model ***P(*** s' **|s ,a)**

Local Variables: U a vectors of utilities for states in ***S***, initially 0

π ,a policy vector indexed by state, initially random

Repeat:

$U \leftarrow Policy - Evaluation\,(\pi, U, mdp)$

$unchanged? \leftarrow true$

For each state **s in *S* do:**

$U'[s] \leftarrow max_a \sum_{s'} P(s' \mid s, a) U(s') > \sum_{s'} P(s' \mid s, a) U(s')$ **then do**

$\pi[s] \leftarrow argmax_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$

Until $unchanged?$

Return π

We implement the `Policy-Iteration` algorithm in our program shown above. π is the optimal policy sequence corresponding to each state. The policy iteration algorithm alternates the following two steps, beginning from some initial policy π_0 :

Policy evaluation: Given a policy π_i, calculate the $U_i = U^{\pi i}$, the utility of each state if π_i were to be executed.

Policy improvement: Calculate a new MEU policy π_{i+1} ,using one-step look-ahead based on U_i. And the algorithm terminates when the policy improvement step yields no change in

the utilities.
Therefore, with the optimal utility results shown above together with the policy results from the program, I find that after $State(Used6)$ the utility is unchanged. Therefore, as a bot, the optimal policy after that State should be replaced. Because, So, we can easily get the best action in each state.

$P_*(New)$ = Use
$P_*(Used_1)$ = Use
$P_*(Used_2)$ = Use
$P_*(Used_3)$ = Use
$P_*(Used_4)$ = Use
$P_*(Used_5)$ = Use
$P_*(Used_6)$ = Replace
$P_*(Used_7)$ = Replace
$P_*(Used_8)$ = Replace
$P_*(Dead)$ = Replace

c) Instead of buying a new machine, a MachineSellingBot offers you the following option: you could buy a used machine, which had an equal chance of being in Used1 and Used2. If the MachineSellingBot were offering you this option for free, you would never buy a new machine. If the MachineSellingBot were offering you this option at a cost of 250, you would never take this option over buying a new machine. What is the highest price for which this used machine option would be the rational choice?

According to the optimal utility results shown in b), in State S, if I want to replace it with a new machine, the $Profit(New)$ is equal to
$U_*(New) - cost\ for\ replace(New) = 800.5 - 250 \approx 550.5$. Therefore, if I want to replace it with a $Used_1\ or\ Used_2$ machine, the $Profit(Used_1)\ or\ Profit(Used_2)$ should be bigger than $Profit(New)$. And only in this way, the customer are willing to buy $Used1\ or\ Used2$ machines other than New machines. Therefore, both
$U_*(Used_1) - cost\ for\ replace(Used_1) = Profit(Used_1) \geqslant 550.5$
$U_*(Used_2) - cost\ for\ replace(Used_2) = Profit(Used_2) \geqslant 550.5$
Therefore, $227.86 \geqslant cost\ for\ replace(Used_1)\ and\ 92.7 \geqslant cost\ for\ replace(Used_2)$. Also it has an equal chance of being in $Used_1$ and $Used_2$, so the final price for $Used_1$ and $Used_2$ should be the average price between the price of $Used_1$ and the price of $Used_2$, which should be 160.28.

d) For different values of β (such that 0 < β < 1), the utility or value of being in certain states will change. However, the optimal policy may not. Compare the optimal policy for β = 0.1, 0.3, 0.5, 0.7, 0.9, 0.99, etc. Is there a policy that is optimal for all sufficiently large β? What do you make of it?

For the problem, I run my program with β = 0.1, 0.3, 0.5, 0.7, 0.9, 0.99,0.999,0.9999. And analyze the results for optimal policy for all sufficiently large β. According to the tables below, we can find that there is an optimal for all sufficiently large β which is shown below. And after $Used_4$ state, replace operation will give us more profit.

$P_*(New)$ = Use

$P_*(Used_1)$ = Use

$P_*(Used_2)$ = Use

$P_*(Used_3)$ = Use

$P_*(Used_4)$ = Replace

$P_*(Used_5)$ = Replace

$P_*(Used_6)$ = Replace

$P_*(Used_7)$ = Replace

$P_*(Used_8)$ = Replace

$P_*(Dead)$ = Replace

	β = 0.1	β = 0.3	β = 0.5	β = 0.7	β = 0.9	β = 0.99	β = 0.999	β = 0.9999
$U_*(New)$	109.9	138.3	188.8	302.9	800.5	6943.95	68176.7	680472.1
$U_*(Used_1)$	99.8	127.9	177.7	289.8	778.3	6913.08	68144.8	680440.1
$U_*(Used_2)$	88.6	113.0	155.5	246.5	643.2	6702.2	67926.0	680220.5
$U_*(Used_3)$	77.4	98.0	133.0	203.4	556.1	6636.75	67865.6	680160.7
$U_*(Used_4)$	66.1	83.0	110.0	160.6	502.8	6624.51	67858.5	680154.0
$U_*(Used_5)$	54.9	67.7	85.3	118.4	475.8	6624.51	67858.5	680154.0
$U_*(Used_6)$	43.6	50.6	55.9	77.1	470.4	6624.51	67858.5	680154.0
$U_*(Used_7)$	30.9	25.5	15.9	37.0	470.4	6624.51	67858.5	680154.0
$U_*(Used_8)$	0.897	-31.9	-46.9	-1.45	470.4	6624.51	67858.5	680154.0
$U_*(Dead)$	-239.0	-208.4	-155.5	-37.9	470.4	6624.51	67858.5	680154.0

	β = 0.1	β = 0.3	β = 0.5	β = 0.7	β = 0.9	β = 0.99	β = 0.999	β = 0.9999
$P_*(New)$	Use	Use	Use	Use	Use	Use	Use	Use
$P_*(Used_1$	Use	Use	Use	Use	Use	Use	Use	Use
$P_*(Used_2$	Use	Use	Use	Use	Use	Use	Use	Use
$P_*(Used_3$	Use	Use	Use	Use	Use	Use	Use	Use
$P_*(Used_4$	Use	Use	Use	Use	Use	Replace	Replace	Replace
$P_*(Used_5$	Use	Use	Use	Use	Use	Replace	Replace	Replace
$P_*(Used_6$	Use	Use	Use	Use	Replace	Replace	Replace	Replace
$P_*(Used_7$	Use	Use	Use	Use	Replace	Replace	Replace	Replace
$P_*(Used_8$	Use	Use	Use	Use	Replace	Replace	Replace	Replace
$P_*(Dead)$	Replace	Replace	Replace	Replace	Replace	Replace	Replace	Replace

Bonus: The cost of a new machine is 250. What is the (long term discounted) value of a new machine? Determine the break-even cost of a new machine - the price above which you are operating at a net loss, and below which you are operating at a net gain.

According to a), $U_*(New)$ = 800.5316006249429. So, the value of a new machine is 800.5. After iteration with a variety of different new machine cost, I found when the costs 10150, the $U_*(New)$ = 0. It means with that cost, we will make no profit using the new machine. So, 10150 is the break-even *cost of a new machine.*