Homework #3

Masafumi Endo, M.S. Student in Robotics
ROB534 - Sequential Decision Making in Robotics
Oregon State University

March 8, 2021

Questions

a

The state space $\mathcal{S}$, action space $\mathcal{A}$, observation space Ω are expressed as,

$$\mathcal{S} = \{s_{left}, s_{right}\}\,, \tag{1}$$
$$\mathcal{A} = \{a_{left}, a_{right}, a_{listen}\}\,, \tag{2}$$
$$\Omega = \{o_{left}, o_{right}\}\,. \tag{3}$$

The reward function $r(s, a)$ is expressed as Tab. 1.

Table 1: Reward function

	a_{left}	a_{right}	a_{listen}
s_{left}	-100	$+10$	-1
s_{right}	$+10$	-100	-1

The probabilistic state-action transition function $T(s, a, s') = P(s'|s, a)$ is expressed as Tab. 2.

Table 2: Transition function

a_{left}	s_{left}	s_{right}
s'_{left}	0.5	0.5
s'_{right}	0.5	0.5

a_{right}	s_{left}	s_{right}
s'_{left}	0.5	0.5
s'_{right}	0.5	0.5

a_{listen}	s_{left}	s_{right}
s'_{left}	1.0	0.0
s'_{right}	0.0	1.0

The conditional observation probabilities function $O(s', a, o) = P(o|a, s')$ is expressed as Tab. 3. Note that for a_{left} and a_{right}, there are no specific description about the observation. Hence, I decided the observation probabilities are equally distributed in $O(s', a_{left}, o)$ and $O(s', a_{right}, o)$.

Table 3: Observation function

a_{left}	s'_{left}	s'_{right}
o_{left}	0.5	0.5
o_{right}	0.5	0.5

a_{right}	s'_{left}	s'_{right}
o_{left}	0.5	0.5
o_{right}	0.5	0.5

a_{listen}	s'_{left}	s'_{right}
o_{left}	0.75	0.25
o_{right}	0.25	0.75

b

The horizon 1 value function is shown in Fig. 1.

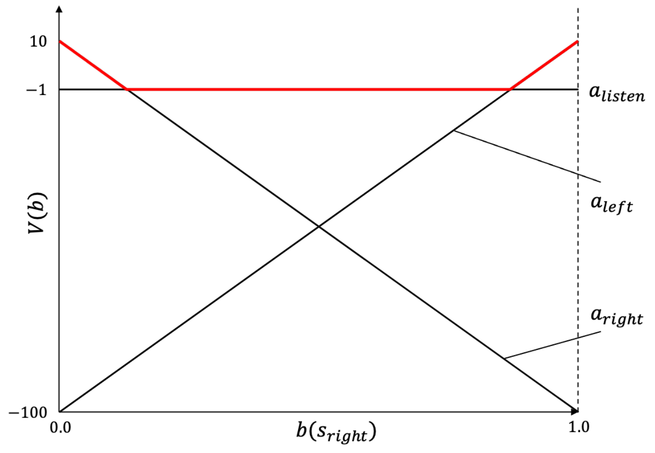


Figure 1: The horizon 1 value function. X-axis denotes belief of s_{right} and y-axis denotes expected rewards. Black lines express α vectors for each action and red line expresses piece-wise linear convex function.

c

The new probability that the tiger is behind the right door is calculated as,

$$\begin{aligned} b'(s'_{right}) &= \frac{P(o_{left}|s'_{right}, a_{listen}) \sum_{s\in\mathcal{S}} P(s'_{right}|s, a_{listen})b(s)}{\sum_{s\in\mathcal{S}, s'\in\mathcal{S}} P(o_{left}|s', a_{listen})P(s'|s, a_{listen})b(s)} && (4) \\ &= 0.25 \times (1.0 \times 0.75 + 0.0 \times 0.25)/0.375 && (5) \\ &= 0.5, && (6) \end{aligned}$$

when I believe the tiger is behind the right with probability 0.75 then perform the listen action and hear that the tiger is on the left.
Same as the above procedure, the following circumstances are treated as,

$$\begin{aligned} b'(s'_{right}) &= \frac{P(o_{right}|s'_{right}, a_{listen}) \sum_{s\in\mathcal{S}} P(s'_{right}|s, a_{listen})b(s)}{\sum_{s\in\mathcal{S}, s'\in\mathcal{S}} P(o_{right}|s', a_{listen})P(s'|s, a_{listen})b(s)} && (7) \\ &= 0.75 \times (1.0 \times 0.75 + 0.0 \times 0.25)/0.625 && (8) \\ &= 0.9, && (9) \end{aligned}$$

when I heard the tiger on the right instead of the left,

$$\begin{aligned} b'(s'_{right}) &= \frac{P(o_{left}|s'_{right}, a_{right}) \sum_{s\in\mathcal{S}} P(s'_{right}|s, a_{right})b(s)}{\sum_{s\in\mathcal{S}, s'\in\mathcal{S}} P(o_{left}|s', a_{right})P(s'|s, a_{right})b(s)} && (10) \\ &= 0.5 \times (0.5 \times 0.75 + 0.5 \times 0.25)/0.5 && (11) \\ &= 0.5, && (12) \end{aligned}$$

when I performed the open-right action instead of listening, and

$$\begin{aligned} b'(s'_{right}) &= \frac{P(o_{left}|s'_{right}, a_{listen}) \sum_{s\in\mathcal{S}} P(s'_{right}|s, a_{listen}) b(s)}{\sum_{s\in\mathcal{S}, s'\in\mathcal{S}} P(o_{left}|s', a_{listen}) P(s'|s, a_{listen}) b(s)} && (13) \\ &= 0.25 \times (1.0 \times 0.3333 + 0.0 \times 0.6667)/0.58335 && (14) \\ &= 0.1428, && (15) \end{aligned}$$

when the initial belief was 0.3333 instead of 0.75.

Programming Assignment

Step 1

The value iteration is executed w/ discount factor $\gamma = 0.9$, Bellman error threshold $\epsilon = 0.00001$. The noise value is used as 0.1 or 0.2.

a

The list of states and rewards are shown in Tab. 4. Note that states are represented as indices of the map environment obtained from positional information. The list of actions are shown in Tab. 5.

Table 4: List of state-reward pairs

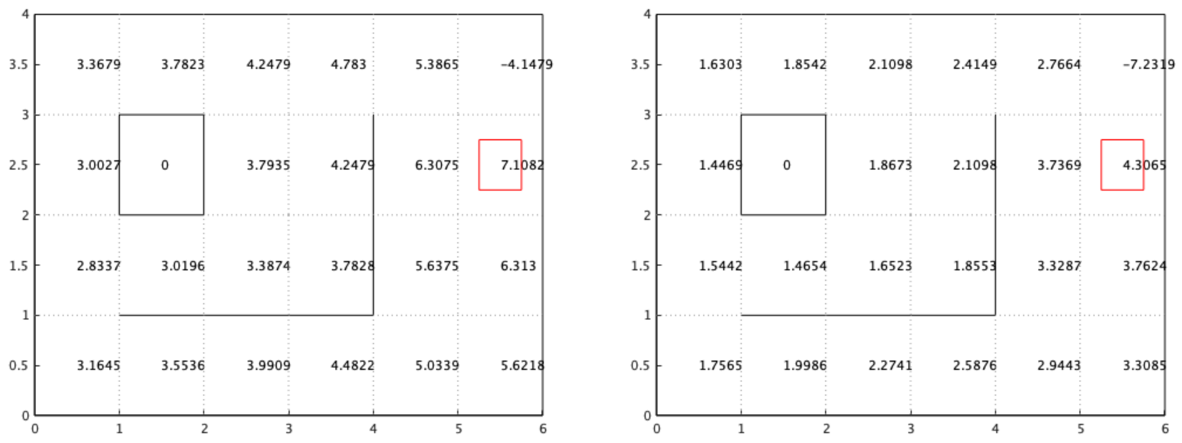
i	1	2	3	4	5	6	7	8	9	10	11	12
s_i	1	2	3	4	5	6	7	8	9	10	11	12
r_i	0	0	0	0	0	0	0	0	0	0	0	0
i	13	14	15	16	17	18	19	20	21	22	23	24
s_i	13	14	15	16	17	18	19	20	21	22	23	24
r_i	0	0	0	0	0	0	0	0	-10	1	0	0

Table 5: List of actions

i	1	2	3	4
a_i	move north	move east	move south	move west

b

Figure 2 shows the learned value function for (a) noises 0.1 and (b) 0.2.



(a) (b)

Figure 2: The learned value function w/ (a) noises 0.1 and (b) 0.2 conditions. The red box represents the location of the robot after 100 iterations.

c

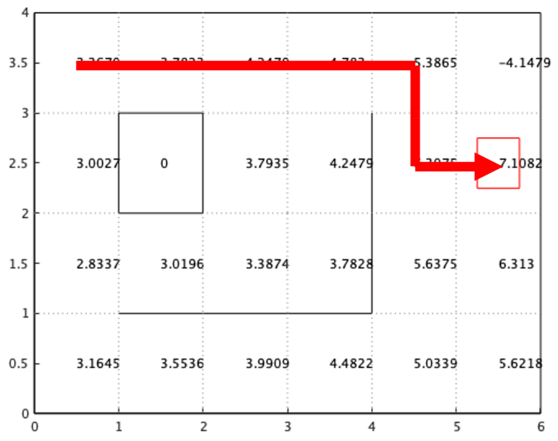
The final reward found by running the algorithm for 100 iteration is shown in Tab. 6.

Table 6: Final reward

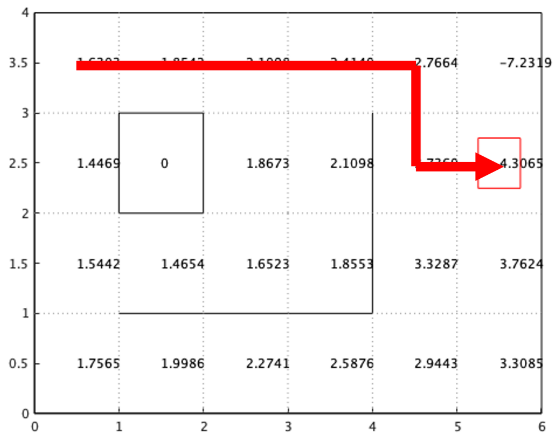
noise	final reward
0.1	79
0.2	23

d

While the optimal policies from the fixed starting point towards a goal position do not change, as shown in Fig. 3, noise of executing actions affects learned Q-values. As shown in Tab. 6, the more noise added, the less rewards are obtained since the robot more frequently be at a state with negative reward next to the goal position. In addition, the policy with different action noises will be changed if I relax ϵ condition. If the noise is 0.1, the value iteration process quickly finds optimal policy since there are few chances for the robot to wrongly execute its actions. On the other hand, if the noise is 0.2, the policy based on value iteration might not be the optimal one since the larger ϵ stops its process before convergence.



(a)



(b)

Figure 3: The optimal policy obtained from the learned value function w/ (a) noises 0.1 and (b) 0.2 conditions. The red box represents the location of the robot after 100 iterations.

Step 2

This section implements Q_{MDP} value method and most-likely search method. For both approaches, value iteration is necessary to provide Q-function. The value iteration is executed w/ discount factor $\gamma = 0.9$, Bellman error threshold $\epsilon = 0.00001$. The noise value is used as 0.0 (no noise) or 0.3.

a

The state space is expressed as following matrix.

$$\mathcal{S} = \begin{pmatrix} s_{11} & \dots & s_{1j} & \dots & s_{1J} \\ \vdots & \ddots & & & \vdots \\ s_{i1} & & s_{ij} & & s_{iJ} \\ \vdots & & & \ddots & \vdots \\ s_{I1} & \dots & s_{Ij} & \dots & s_{IJ} \end{pmatrix}$$

The subscripts i $(= 1 \dots 24)$ express the state space's index for the robot and j $(= 1 \dots 24)$ express the state space's index for the target. The list of actions is expressed as following matrix.

$$\mathcal{A} = \begin{pmatrix} \{a_1^{robot}, a_1^{target}\} & \{a_1^{robot}, a_2^{target}\} & \{a_1^{robot}, a_3^{target}\} & \{a_1^{robot}, a_4^{target}\} \\ \{a_2^{robot}, a_1^{target}\} & \{a_2^{robot}, a_2^{target}\} & \{a_2^{robot}, a_3^{target}\} & \{a_2^{robot}, a_4^{target}\} \\ \{a_3^{robot}, a_1^{target}\} & \{a_3^{robot}, a_2^{target}\} & \{a_3^{robot}, a_3^{target}\} & \{a_3^{robot}, a_4^{target}\} \\ \{a_4^{robot}, a_1^{target}\} & \{a_4^{robot}, a_2^{target}\} & \{a_4^{robot}, a_3^{target}\} & \{a_4^{robot}, a_4^{target}\} \end{pmatrix}$$

The subscripts of a correspond the action list shown in Tab. 5. The observation function $O(s', o)$ is expressed as follows:

$$O(s', o) = P(o|s') = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

The reward function $R(s)$ is expressed as follows:

$$R(s) = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

b

The belief update process has two main steps: motion update (prediction step) and observation update (update process). Motion update calculate belief $\bar{b}(s')$ prior to observe the target as follows,

$$\bar{b}(s') = \sum_{s \in \mathcal{S}} P(s'|s) b(s), \tag{16}$$

where $P(s'|s)$ express state transition probability function for the target to move the environment without any noise. Observation update calculate new belief $b'(s')$ after observing the target as follows,

$$b'(s') = \eta P(o|s') \bar{b}(s') \tag{17}$$

where η expresses normalization term.

c

The final reward found by running the algorithms for 100 iterations with different noise levels are shown in Tab. 7 and 8.

Table 7: Q_{MDP} search

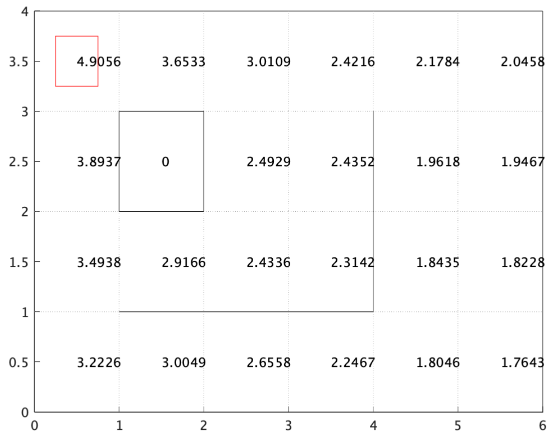
noise	final reward
0.0	15
0.3	12

Table 8: Most-likely search

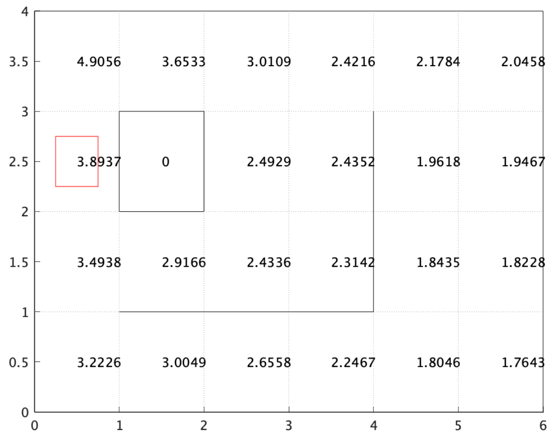
noise	final reward
0.0	22
0.3	12

d

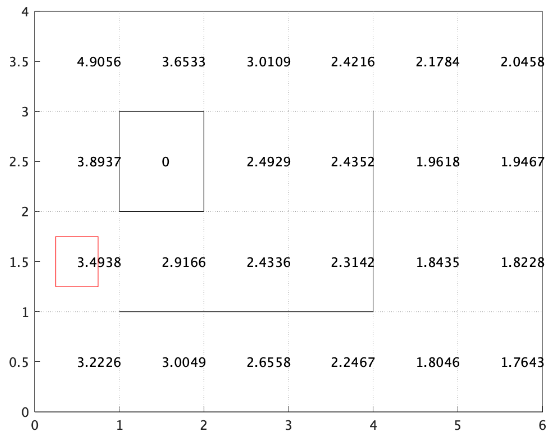
Figure 4 shows four subsequent map frames using the most-likely heuristic at the 0.3 noise level.



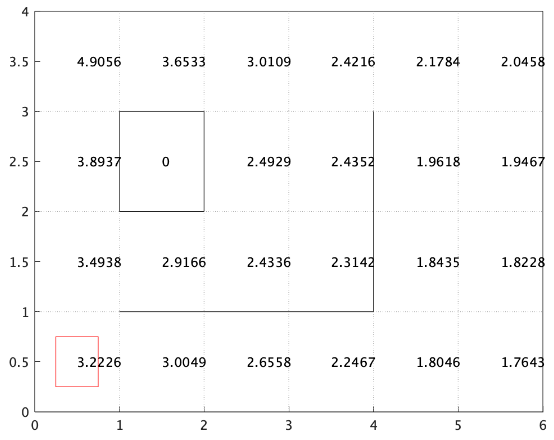
(a) iteration 1



(b) iteration 2



(c) iteration 3

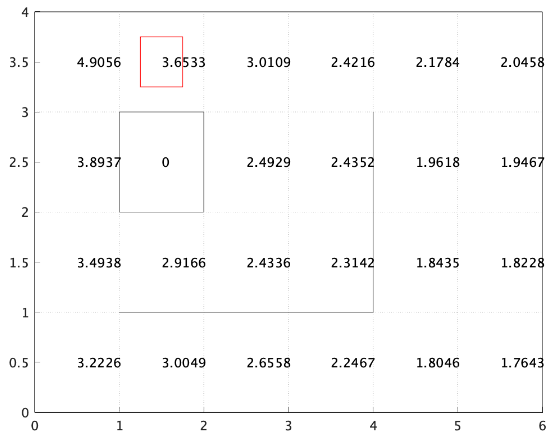


(d) iteration 4

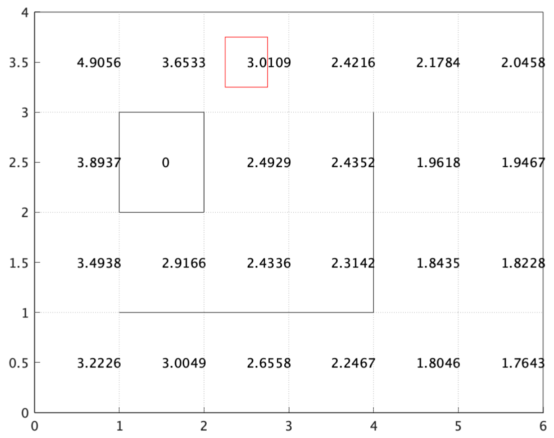
Figure 4: Four subsequent image frames using the most-likely heuristic at the 0.3 noise level.

e

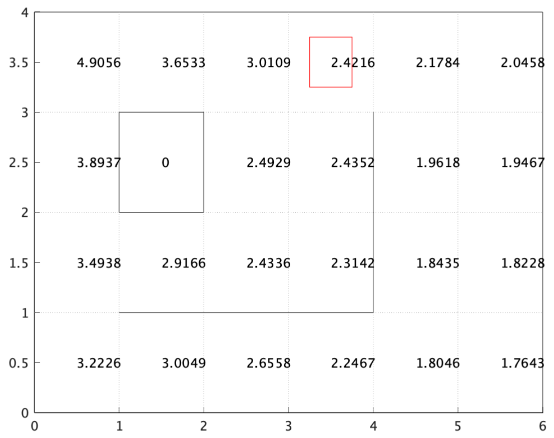
Figure 4 shows four subsequent map frames using $\mathrm{Q_{MDP}}$ at the 0.3 noise level.



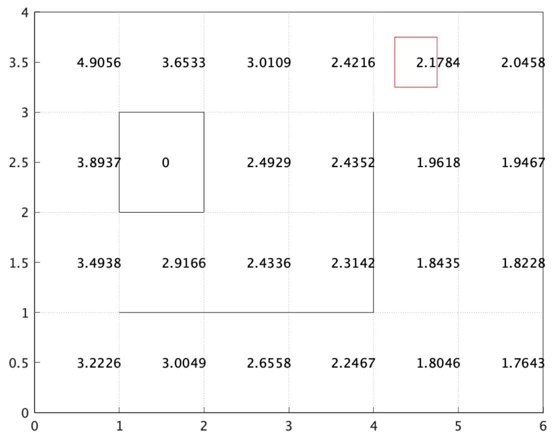
(a) iteration 1



(b) iteration 2



(c) iteration 3



(d) iteration 4

Figure 5: Four subsequent image frames using $\mathrm{Q_{MDP}}$ at the 0.3 noise level.

Discussion Questions

1

Algorithm 1 and 2 show algorithmic descriptions of $\mathrm{Q_{MDP}}$ search and most-likely search. The former heuristic uses belief to calculate Q_{MDP}, then finds the best action to get maximal Q_{MDP} value. The later heuristic, on the other hand, identifies target state based on maximal belief then directly use Q-function to find the best action. In other words, $\mathrm{Q_{MDP}}$ heuristic adjust Q-function to consider uncertain target's movement using belief space, while most-likely search greedily search target position based on belief space. Most-likely heuristic would guide the reward position more directly so that it is a more reasonable heuristic compared to $\mathrm{Q_{MDP}}$ to solve this POMDP. As shown in Tab. 7 and 8, the behavior of most-likely search can gather more rewards.

Algorithm 1 $\mathrm{Q_{MDP}}$ search

```
for $i$=1 to 100 do
    $\overline{bel} = T_{bel} bel$
    for $a \in \mathcal{A}$ do
        $Q_{\mathrm{MDP}}(a) = \sum_{s^{target} \in \mathcal{S}^{target}} \overline{bel}(s^{target}) Q(a, s^{target})$
    end for
    $a^* = \arg\max_{a \in \mathcal{A}} Q_{\mathrm{MDP}}(a)$
    $s'^{robot}$ = MoveMaze($s^{robot}, a^*$)
    $obs$ = GetObservation($s'^{robot}$)
    $bel$ = ObservationUpdate($obs, \overline{bel}, s'^{robot}$)
end for
```

Algorithm 2 Most-likely search

```
for $i$=1 to 100 do
    $\overline{bel} = T_{bel} bel$
    $s^{target} = \arg\max_{s^{target} \in \mathcal{S}^{target}} \overline{bel}(s^{target})$
    $a^* = \arg\max_{a \in \mathcal{A}} Q(s^{target}, a)$
    $s'^{robot}$ = MoveMaze($s^{robot}, a^*$)
    $obs$ = GetObservation($s'^{robot}$)
    $bel$ = ObservationUpdate($obs, \overline{bel}, s'^{robot}$)
end for
```

2

Suppose the robot tries to reach static goal locations while it is operated GPS-denied environment. It has to locally perceive surrounding environment for localize itself, and I formulate this situation as POMDP. The robot cannot perfectly observe environment so there is uncertainty about its state. In such case, $\mathrm{Q_{MDP}}$ solver relatively works well since it gradually update belief space to optimize Q-function and get optimal policy. Most-likely search, on the other hand, greedily get maximal belief state so it would fall into local optima.

3

POMDP solvers will provide better trajectories for the robot to actively minimize observation uncertainty, while the two approaches in this assignment only “reflect” uncertainty for its policy but the robot doesn’t act for minimizing it. It is the advantage of more sophisticated POMDP solvers. However, to solve POMDP, more computational cost is necessary due to combinatorial possible state transition. Hence, we need to consider ways of approximation, sampling, or generalization techniques to solve POMDP.