

CPSC 502 — Fall 2006 Non-Assignment 5

This is not an assignment in the sense that it will not be marked. I will, however, assume that you have done it when I am setting the midterm. A solution will be posted.

Question 1

Consider the planning domain in the textbook.

- Give the feature-based representation of the *MW* and *RHM* features.
- Give the STRIPS representations for the pickup mail, and the deliver mail actions.
- What are the errors on lecture 11.2, page 4?
- What are the errors on lecture 11.3, page 5?
- Suppose the robot cannot carry both coffee and mail at the same time. Give two different ways that the CSP that represents the planning problem can be changed to reflect this. Test it, by giving a problem where the answer is different when the robot has this limitation than when it doesn't.

Solution

- The *MW* feature can be axiomatized by defining when $MW = true$ (written as mw) and using negation as failure:

$$mw' \leftarrow mw \wedge Action \neq pum$$

The *RHM* feature can be axiomatized by defining when $RHM = true$ (written as rhm) and using negation as failure:

$$rhm' \leftarrow Action = pum$$

$$rhm' \leftarrow rhm \wedge Action \neq dm$$

- The pickup mail action (pum) is defined using STRIPS by:

Preconditions: $RLoc = mr \wedge mw$

Effects: $[\overline{mw}, rhm]$

The deliver mail action (dm) is defined using STRIPS by:

Preconditions: $RLoc = off \wedge rhm$

Effects: $[\overline{rhm}]$

- Errors in Lecture 11.2, “What are the errors?” slide:

- (2) should have $MW = false$ (i.e., \overline{mw}).
- (4) should not exist as the preconditions of puc are not satisfied in (1).
- There should be a neighbour of (1) for the mac action.
- In (5), the robot shouldn't get coffee and drop mail
- In (6), the location should be *off*.
- in (7), Sam should still want coffee after the robot picks up coffee.
- There should be a neighbour of (3) for mac .
- In (11) the robot should have coffee.

- (10) should have neighbors for mc and mac .
- (d) Errors in Lecture 11.3, “Find the errors” slide:
- In (2) off should be mr (as the precondition of picking up mail is that the robot is in the mail room).
 - (3) shouldn't exist, as rhc cannot be true immediately after dc .
 - in (7), there should be rhc , not \overline{rhc} .
 - (8) should not exist as a prerequisite for puc is that the robot is in the coffee shop and it can't be in the coffee shop and the office at the same time.
 - (2) should have a child for the mac action (to achieve off).
 - (6) shouldn't exist as dm doesn't achieve any goal
- (e)
- You can make a constraint between the state variables RHC and RHM .
 - You can change the precondition of the action puc and pum so that puc has \overline{rhm} as a precondition and pum has rhc as a precondition.

Question 2

These questions are available from the textbook available on the course WebCT site.

- (a) Do exercise 12.2
- (b) Do exercise 12.4
- (c) Do exercise 12.5

Solution

- (a) You can get a CIspace representation of a solution for the student who just wants to pass from <http://www.cs.ubc.ca/spider/poole/ci2/figures/ch12/StudyDecision2.xml> or a representation for a student who wants to really do well from <http://www.cs.ubc.ca/spider/poole/ci2/figures/ch12/StudyDecision1.xml>
- (b)
- After eliminating run by maximizing, you have the following factor on $look$ and see :

look	see	value
true	true	23
true	false	56
false	true	28
false	false	22

- The optimal decision function for run is:

look	see	run
true	true	yes
true	false	no
false	true	yes
false	false	no

That is, if the agent sees, it should run.

- (c) The value of “positive test” for the decision “discard sample” is positive (greater than zero). The value of “contaminated specimen” for the decision “discard sample” is zero.

Question 3

1. Suppose our Q-learning agent, with fixed alpha (α), and discount gamma (γ), was in state 34 did action 7, received reward 3 and ended up in state 65. What value(s) get updated? Give an expression for the new value. (You need to be as specific as possible)
2. In temporal difference learning (e.g. Q-learning), to get the average of a sequence of k values, we let $\alpha_k = 1/k$. Explain why it may be advantageous to keep alpha fixed in the context of reinforcement learning.
3. Suppose someone suggested using $\alpha_k = 10.0/(9.0 + k)$. Explain why it is of this form (e.g., why $9 + k$ on the bottom?) Would you expect it to work well? Explain.
4. Explain what happens in reinforcement learning if the agent always chooses the action that maximizes the Q-value. Suggest two ways that can force the agent to explore.
5. In MDPs and reinforcement learning explain why we often use discounting of future rewards. How would an agent act differently if the discount factor was 0.6 as opposed to 0.9.
6. What is the main difference between asynchronous value iteration and standard value iteration? Why does asynchronous value iteration often work better than standard value iteration?
7. What is the relationship between asynchronous value iteration and Q-learning?

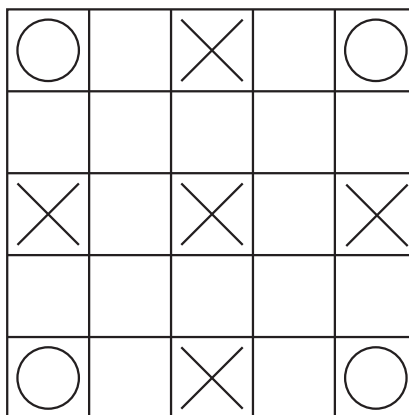
Solution

1. $q[34, 7] = q[34, 7] + \alpha * (3 + \gamma * \max_a q[65, a] - q[34, 7])$
2. Using $\alpha_k = 1/k$ means all estimates are treated equally, whereas having α_k fixed means that more recent values have a higher weight. In particular, when α_k is fixed it approaches quicker to approximately correct values, but it doesn't converge unless the system is deterministic. [The reason it doesn't converge is because of variability in the reward and the resulting state.] A fixed α_k also allows for adaptation when the environment changes.
3. This is a way to combine all of the examples, but still count more recent data more old data. This does converge to the actual Q-values. It is of this form so that when $k = 1$, it have value 1 (for the first data point it uses the actual value). This converges much quicker than using $1/k$ or a fixed alpha. See <http://www.cs.ubc.ca/spider/poole/demos/rl/q10.html> Check which converges to the same values as the value iteration applet.
4. Another action may be better, but this is never discovered because that action is not tried. The two main ways to explore: include some random actions or to initialize with high Q-values so that unexplored areas of the search space look good.
5. Discounting is used to make sure the values are finite and to make future rewards less valuable than immediate rewards. This encourages it to find a short path than a long path. If the discount was 0.6, rewards further in the future are not worth as much as they are when the reward is 0.9. Thus the agent will act more for immediate rewards. (Try the value iteration applet and change the discount factor.)
6. Asynchronous value iteration allows the values to be updated in any order. It works better because always uses the latest values, and it can concentrate on the areas where the values change the most. There is only need for one array, not two that value iteration needs.

7. Q-learning is using asynchronous value iteration, but is using its experience rather than the model to get the expected value. The values that it updates are the state-action pairs the agent actually visits.

Question 4

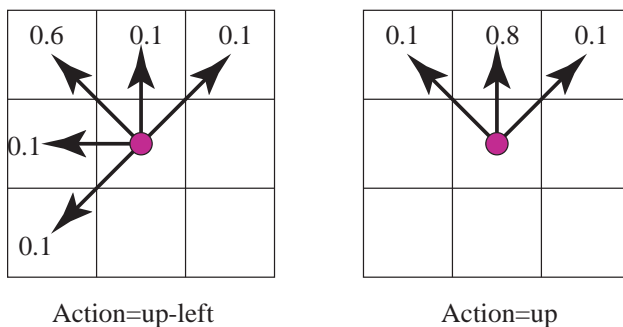
Consider the game domain:



The robot can be at one of the 25 locations on the grid. There can be a treasure on one of the circles at the corners. When the robot reaches the corner where the treasure is, it collects a reward of 10, and the treasure disappears. When there is no treasure, at each time step, there is a probability $P_1 = 0.2$ that a treasure appears, and it appears with equal probability at each corner. The robot knows its position and where the treasure is.

There are monsters at the squares marked with an X. Each monster randomly and independently, at each time, checks if the robot is on their square. If the robot is on the square when the monster checks, it has a reward of -10 (i.e., it loses 10 points). At the centre point the monster checks at each time with probability $p_2 = 0.4$, at the other 4 squares marked with an X, the monsters check at each time with probability $p_3 = 0.2$.

The robot has 8 actions corresponding to the 8 neighbouring squares. The diagonal moves are noisy; there is a $p_4 = 0.6$ probability of going in the direction chosen and an equal chance of going to each of the 4 neighboring squares that are closest to the desired direction. The vertical and horizontal moves are also noisy; there is a probability $p_5 = 0.8$ chance of going in the requested direction and an equal chance of going to one on the adjacent diagonal squares. For example, the actions up-left and up have the following result:



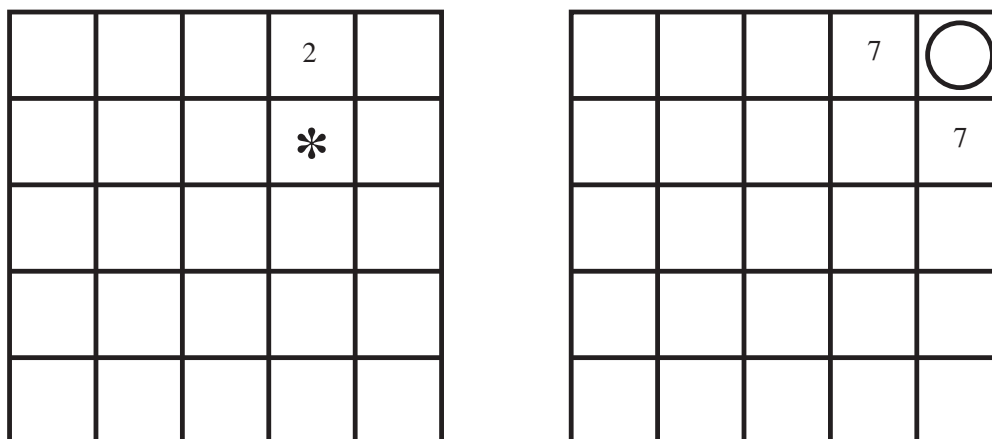
If the action would result in crashing in a wall, the robot has a reward of -2 (i.e., loses 2) and does not move.

There is a discount factor of $p_6 = 0.9$.

Assume that the rewards are immediate on entering a state (i.e., if the robot enters a state where there is a monster, it gets the (negative) reward on entering the state, and if the robot enters the state where there is a treasure it gets the reward on entering the state, even if the treasure arrives at the same time).

(a) Suppose we are using the inefficient state space representation with 125 states.

Suppose we are running value iteration, and have the following values for each state: [The numbers in the square represent the value of that state, and where empty squares have a zero value. It is irrelevant to this question how these values got there]:



where the treasure is at the circle. There are also states for the treasures at the other three corners (assume that the current Q-values for these states are all zero).

Consider the next step of value iteration. For state s_{13} , which is marked by * in the above figure, and the action a_2 which is “up”, what value is assigned to $Q[s_{13}, a_2]$ on the next iteration of value iteration? You need to show all working, but don't need to do any arithmetic (i.e., leave it as an expression). Explain each terms in your expression.

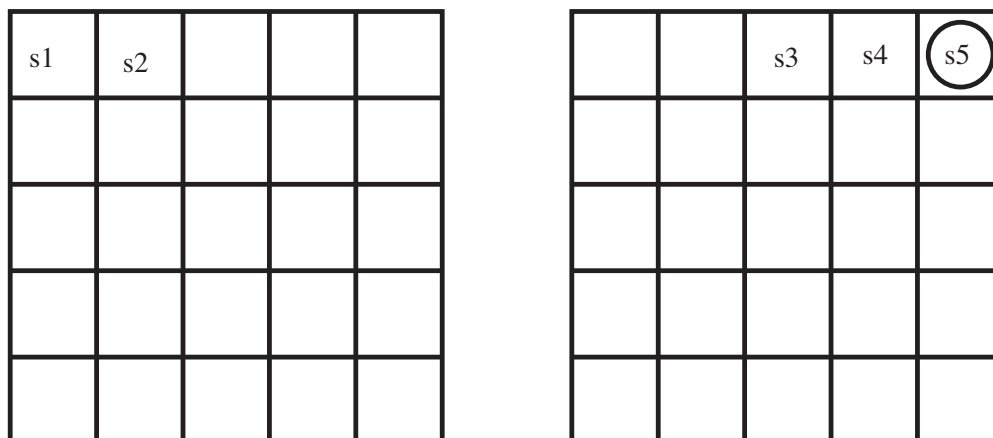
Solution

$$\begin{aligned}
 Q[s_{13}, a_2] = & \\
 & 0.8 * 0.8 * (0 + 0.9 * 2) \quad \text{— up, no treasure} \\
 + & 0.8 * 0.2 * 0.25 * (0 + 0.9 * 7) \quad \text{— up, treasure at top right} \\
 + & 0.1 * 0.8 * (0.2 * -10 + 0.9 * 0) \quad \text{— left, no treasure} \\
 + & 0.1 * 0.2 * (0.2 * -10 + 0.9 * 0) \quad \text{— left, treasure appears} \\
 + & 0.1 * 0.2 * 0.25(10 + 0.9 * 0) \quad \text{— right, treasure appears there}
 \end{aligned}$$

every other value is 0. Note that $0.2 * 0.25$ is the probability that a treasure appears at the top right state.

Question 5

Consider the same domain, but where the agent isn't given a model. Suppose that the agent steps through the state space in the order of steps given in the diagram below, (i.e., going from s_1 to s_2 to s_3 to s_4 to s_5), each time doing a “right” action.



Note that in this figure, the numbers represent the order that the robot visited the states. You can assume that this is the first time the robot has visited any of these states.

- Suppose a monster did not appear at any time during any of these experiences. What Q-values are updated during Q-learning based on this experience? Explain what values they get assigned. You should assume that $\alpha_k = 1/k$.
- Suppose that, at some later time, the robot revisits the same states: $s1$ to $s2$ to $s3$ to $s4$ to $s5$, and hasn't visited any of these states in between (i.e, this is the second time visiting any of these states). Suppose this time, the monster appears so that the robot gets a penalty. What Q-values have their values changed? What are their new values?

Solution

- $Q[s4, right] = 10$
- $Q[s2, right] = -10/2 = -5.$
 $Q[s3, right] = 0.9 * 10/2 = 4.5.$
 $Q[s4, right] = 10 + 0.5 * (10 - 10) = 10.$