CS 188
Spring 2014 — Introduction to Artificial Intelligence

Written HW5

INSTRUCTIONS

- **Due:** Monday, March 3rd 2014 11:59 PM
- **Policy:** Can be solved in groups (acknowledge collaborators) but must be written up individually. However, we strongly encourage you to first work alone for about 30 minutes total in order to simulate an exam environment. Late homework will not be accepted.
- **Format:** You must solve the questions on this handout (either through a pdf annotator, or by printing, then scanning; we recommend the latter to match exam setting). Alternatively, you can typeset a pdf on your own that has answers appearing in the same space (check edx/piazza for latex templating files and instructions). **Make sure that your answers (typed or handwritten) are within the dedicated regions for each question/part. If you do not follow this format, we may deduct points.**
- **How to submit:** Go to www.pandagrader.com. Log in and click on the class CS188 Spring 2014. Click on the submission titled Written HW 4 and upload your pdf containing your answers. If this is your first time using pandagrader, you will have to set your password before logging in the first time. To do so, click on "Forgot your password" on the login page, and enter your email address on file with the registrar's office (usually your @berkeley.edu email address). You will then receive an email with a link to reset your password.

Last Name	Chen
First Name	Jianzhong
SID	23478230
Email	chenjianzhong@berkeley.edu
Collaborators	None

For staff use only

Q. 1	Q. 2	Total
/22	/8	/30

Q1. [22 pts] Worst-Case Markov Decision Processes

Most techniques for Markov Decision Processes focus on calculating $V^*(s)$, the maximum expected utility of state s (the expected discounted sum of rewards accumulated when starting from state s and acting optimally). This maximum expected utility $V^*(s)$ satisfies the following recursive expression, known as the Bellman Optimality Equation:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right].$$

In this question, instead of measuring the quality of a policy by its expected utility, we will consider the worst-case utility as our measure of quality. Concretely, $L^\pi(s)$ is the minimum utility it is possible to attain over all (potentially infinite) state-action sequences that can result from executing the policy π starting from state s. $L^*(s) = \max_\pi L^\pi(s)$ is the optimal worst-case utility. In words, $L^*(s)$ is the *greatest lower bound* on the utility of state s: the discounted sum of rewards that an agent acting optimally is guaranteed to achieve when starting in state s.

Let $C(s,a)$ be the set of all states that the agent has a non-zero probability of transferring to from state s using action a. Formally, $C(s,a) = \{s' \mid T(s,a,s') > 0\}$. This notation may be useful to you.

(a) [5 pts] Express $L^*(s)$ in a recursive form similar to the Bellman Optimality Equation. Circle all that apply.

● $L^*(s) = \max_a \min_{s' \in C(s,a)} [R(s,a,s') + \gamma * L^*(s')]$

○ $L^*(s) = \min_a \max_{s' \in C(s,a)} [R(s,a,s') + \gamma * L^*(s')]$

○ $L^*(s) = \max_a \sum_{s' \in C(s,a)} T(s,a,s') [R(s,a,s') + \gamma * \min_{s'' \in C(s',a')} L^*(s'')]$

○ $L^*(s) = \max_a [R(s,a,s') + \gamma * \min_{s' \in C(s,a)} L^*(s')]$

○ None of the Above: $L^*(s) =$

(b) [3 pts] Recall that the Bellman update for value iteration is:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

Formally define a similar update for calculating $L_{i+1}(s)$ using L_i.

$L_{i+1}(s) = \max_a \min_{s' \in C(s,a)} [R(s,a,s') + \gamma * L_i(s')]$

(c) [5 pts] From this point on, you can assume that $R(s, a, s') = R(s)$ (rewards are a function of the current state) and that $R(s) \geq 0$ for all s. With these assumptions, the Bellman Optimality Equation for Q-functions is

$$Q^*(s,a) = R(s) + \sum_{s'} T(s,a,s') \left[\gamma \max_{a'} Q^*(s',a')\right]$$

Let $M(s, a)$ be the *greatest lower bound* on the utility of state s when taking action a In words, if an agent plays optimally after taking action a from state s, this is the utility the agent is guaranteed to achieve. Formally define $M^*(s, a)$, in a recursive form similar to how Q^* is defined. Circle all that apply.

○ $M^*(s,a) = R(s) + \min_{s' \in C(s,a)}[R(s') + \gamma \max_{a'} M^*(s',a')]$

○ $M^*(s,a) = R(s) + \sum_{s' \in C(s,a)} T(s,a,s')[R(s) + \gamma \min_{a'} M^*(s',a')]$

○ $M^*(s,a) = \min_{s' \in C(s,a)}[R(s') + \gamma max_{a'} M^*(s',a')]$

○ $M^*(s,a) = \max_{s' \in C(s,a)}[R(s) + \gamma min_{a'} M^*(s',a')]$

● None of the Above: $M^*(s,a) = R(s) + min_{s' \in C(s,a)}[\gamma * max_{a'} M^*(s',a')]$

(d) [5 pts] Recall that the Q-learning update for maximizing expected utility is:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left(R(s) + \gamma \max_{a'} Q(s',a')\right),$$

where α is the learning rate, $(s, a, s', R(s))$ is the sample that was just experienced ("we were in state s, we took action a, we ended up in state s', and we received a reward $R(s)$). Circle the update equation below that results in $M(s, a) = M^*(s, a)$ when run sufficiently long under a policy that visits all state-action pairs infinitely often. If more than one of the update equations below achieves this, select the one that would converge more quickly. Note that in this problem, we do not know T or C when starting to learn.

(i) $C(s,a) \leftarrow \{s'\} \cup C(s,a)$ (i.e. add s' to $C(s,a)$)

$$M(s,a) \leftarrow (1-\alpha)M(s,a) + \alpha \left(R(s) + \gamma \sum_{s' \in C(s,a)} \max_{a'} M(s',a')\right)$$

[✓ (ii)] $C(s,a) \leftarrow \{s'\} \cup C(s,a)$ (i.e. add s' to $C(s,a)$)

$$M(s,a) \leftarrow (1-\alpha)M(s,a) + \alpha \left(R(s) + \gamma \min_{s' \in C(s,a)} \max_{a'} M(s',a')\right)$$

(iii) $C(s,a) \leftarrow \{s'\} \cup C(s,a)$ (i.e. add s' to $C(s,a)$)

$$M(s,a) \leftarrow R(s) + \gamma \min_{s' \in C(s,a)} \max_{a'} M(s',a')$$

(iv) $$M(s,a) \leftarrow (1-\alpha)M(s,a) + \alpha \min\left\{M(s,a), R(s) + \gamma \max_{a'} M(s',a')\right\}.$$

(e) [2 pts] Suppose our agent selected actions to maximize $L^*(s)$, and $\gamma = 1$. What non-MDP-related technique from this class would that resemble? (a one word answer will suffice)

1e: minmax

(f) [2 pts] Suppose our agent selected actions to maximize $L_3(s)$ (our estimate of $L^*(s)$ after 3 iterations of our "value-iteration"-like backup in section b) and $\gamma = 1$. What non-MDP-related technique from this class would that resemble? (a brief answer will suffice)

1f: Minimax search with depth limit and evaluation function.

Q2. [8 pts] Reinforcement Learning

Recall that reinforcement learning agents gather tuples of the form $< s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1} >$ to update the value or Q-value function. In both of the following cases, the agent acts at each step as follows: with probability 0.5 it follows a fixed (not necessarily optimal) policy π and otherwise it chooses an action uniformly at random. Assume that in both cases updates are applied infinitely often, state-action pairs are all visited infinitely often, the discount factor satisfies $0 < \gamma < 1$, and learning rates α are all decreased at an appropriate pace.

(a) [4 pts] The Q-learning agent performs the following update:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Will this process converge to the optimal Q-value function? If yes, write "Yes." If not, give an interpretation (in terms of kind of value, optimality, etc.) of what it will converge to, or state that it will not converge:

2a: Yes

(b) [4 pts] Another reinforcement learning algorithm is called SARSA, and it performs the update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Will this process converge to the optimal Q-value function? If yes, write "Yes." If not, give an interpretation (in terms of kind of value, optimality, etc.) of what it will converge to, or state that it will not converge:

2b: No. It will converge to the Q-value function of a certain policy, not necessarily the optimal Q-value function.