CS 234 Midterm – Spring 2016-17

(Do not turn this page until you are instructed to do so!)

Instructions: Please answer the following questions to the best of your ability. You have **80 minutes** to complete the exam. The exam is closed-book, closed-note, closed-internet, etc. However, you may use one letter-sized sheet (front and back) of notes as reference.

All of the intended answers can be written well within the space provided. Good luck!

The following is a statement of the Stanford University Honor Code:

1. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

   (2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

2. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

3. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By signing your name below, you acknowledge that you have abided by the Stanford Honor Code while taking this exam.

Signature: _____

Name: _____

SUNetID: _____

Problem	#1	#2	#3	#4	#5	#6	Total
Maximum	10	12	6	24	16	12	80

1 Markov Decision Process (10 Points)

You are in a Las Vegas casino! You have \$20 for this casino venture and will play until you lose it all or as soon as you double your money (i.e., increase your holding to at least \$40). You can choose to play two slot machines: 1) slot machine A costs \$10 to play and will return \$20 with probability 0.05 and \$0 otherwise; and 2) slot machine B costs \$20 to play and will return \$30 with probability 0.01 and \$0 otherwise. Until you are done, you will choose to play machine A or machine B in each turn. In the space below, provide an MDP that captures the above description. Describe the state space, action space, rewards and transition probabilities. Assume the discount factor $\gamma = 1$.

You can express your solution using equations instead of a table, or a diagram.

Solution:
Example MDP (tables or diagrams containing the same information are equally valid):

States: s_i where $i \in \{0, 10, 20, 30, 40\}$ and s_0 and s_{40} are terminal
Actions: A or B
Rewards: $R(s_i, a, s_j) = j - i$ ($R(s, a, s') = 1$ if $s' = s_{40}$ else 0) would also work

Transition probabilities:
$P(s_{i-10}|s_i, A) = 0.95$
$P(s_{i+10}|s_i, A) = 0.05$
$P(s_{i-20}|s_i, B) = 0.99$
$P(s_{i+10}|s_i, B) = 0.01$

Where the above transitions are only valid for A in states s_{10}, s_{20}, s_{30} and for B in states s_{20}, s_{30}.
All other transitions do not exist (have probability 0).

General discussion:
An accurate MDP model of the casino scenario would have the following:

States must include all necessary information to judge when we would transition to a terminal state (would have run out of money or reached \$40). This means that states will either directly or indirectly refer to how much money you have left (termination conditions should not rely on additional information from rewards).

Actions could be choosing to use machine A or machine B. Some responses included a special third action to express a loop on terminal states, which is also fine.

Rewards can take many different forms but should yield a higher reward when terminating with \$40 than when terminating with \$0. Also, the reward for terminating with \$40 should be the same regardless of how we got there (and equivalently for \$0) because it is the same result in our original scenario. There are different ways to do this: we could make the reward the incremental change in our holdings, or we could set a lump-sum reward on transitions to the terminal states.

Transition Probabilities: A transitions to a state with 10 fewer dollars with 95% probability and a state with 10 more dollars with 5% probability. B transitions to a state with 20 fewer dollars with 99% probability and a state with 10 more dollars with 1% probability. Note that these transitions are only defined for states where you have enough money to pay the up-front cost of playing the slot machine. In particular, action B cannot be chosen in a state with only \$10.

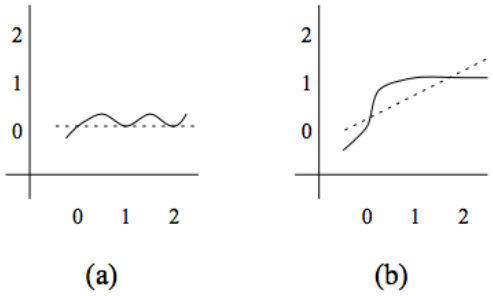
2 Bellman Operator with Function Approximation (12 Points)

Consider an MDP $M = (S, A, R, P, \gamma)$ with finite discrete state space S and action space A. Assume M has dynamics model $P(s'|s, a)$ for all $s, s' \in S$ and $a \in A$ and reward model $R(s, a)$ for all $s \in S$ and $a \in A$.

Recall that the Bellman operator B applied to a function $V : S \to \mathbb{R}$ is defined as

$$B(V)(s) = \max_a (R(s, a) + \gamma \sum_{s'} P(s'|s, a)V(s'))$$

(a) (4 Points) Now, consider a new operator which first applies a Bellman backup and then applies a function approximation, to map the value function back to a space representable by the function approximation. We will consider a linear value function approximator over a continuous state space. Consider the following graphs:



The graphs show linear regression on the sample $X_0 = \{0, 1, 2\}$ for hypothetical underlying functions. On the left, a target function f (solid line), that evaluates to $f(0) = f(1) = f(2) = 0$, and its corresponding fitted function $\hat{f}(x) = 0$. On the right, another target function g (solid line), that evaluates to $g(0) = 0$ and $g(1) = g(2) = 1$, and its fitted function $\hat{g}(x) = \frac{7}{12}x$.

What happens to the distance between points $\{f(0), f(1), f(2)\}$ and $\{g(0), g(1), g(2)\}$ after we do the linear approximation? In other words, compare $\max_{x \in X_0} |f(x) - g(x)|$ and $\max_{x \in X_0} |\hat{f}(x) - \hat{g}(x)|$.

Solution: We compute $\max_{x \in X_0} |f(x) - g(x)| = 1$ and $\max_{x \in X_0} |\hat{f}(x) - \hat{g}(x)| = \frac{7}{6}$. Note $\max_{x \in X_0} |f(x) - g(x)| < \max_{x \in X_0} |\hat{f}(x) - \hat{g}(x)|$. The distance between the points increases after the linear approximation.

(b) (4 Points) Is the linear function approximator here a contraction operator? Explain your answer.

Solution: Let L be the linear approximation operator such that $\hat{f} = Lf$ and $\hat{g} = Lg$. From Part a), we see that $\|Lf - Lg\|_\infty > \|f - g\|_\infty$ where $\|\cdot\|_\infty$ is the infinity norm. Then, the linear function approximator L is not a contraction operator.

(c) (4 Points) Will the new operator be guaranteed to converge to single value function? If yes, will this be the optimal value function for the problem? Justify your answers.

Solution: While the Bellman operator B is a contraction operator, the composite operator $L \circ B$ that first applies a Bellman backup and then a linear approximation is not necessarily a contraction operator because the linear function approximator L is not a contraction operator. Since we do not have the contraction property, the composite operator does not necessarily converge. In fact, there are concrete examples where this operator does not converge.

3 Probably Approximately Correct (6 Points)

Let $A(\alpha, \beta)$ be a hypothetical reinforcement learning algorithm, parametrized in terms of α and β such that for any $\alpha > \beta > 1$, it selects action a for state s satisfying $|Q(s,a) - V^*(s)| \leq \frac{\beta}{\alpha}$ in all but $N = \frac{|S||A|\alpha\beta}{1-\gamma}$ steps with probability at least $1 - \frac{1}{\beta^2}$.

Per the definition of *Probably Approximately Correct Reinforcement Learning*, express N as a function of $|S|$, $|A|$, δ, ϵ and γ. What is the resulting N? Is algorithm A probably approximately correct (PAC)? Briefly justify.

Solution: We want to achieve the bound that $|Q(s,a) - V^*(S)| \leq \epsilon$ with probability $1 - \delta$.

So let $\frac{\beta}{\alpha} = \epsilon$ and $1 - \frac{1}{\beta^2} = 1 - \delta$, which gives $\alpha = \frac{1}{\epsilon\sqrt{\delta}}$ and $\beta = \frac{1}{\sqrt{\delta}}$.

Substituting, $N = \frac{|S||A|\alpha\beta}{1-\gamma} = \frac{|S||A|}{\epsilon\delta(1-\gamma)}$

Since N is a polynomial function of $|S|$, $|A|$, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ and achieves the ϵ, δ bounds above, then A is PAC.

4 Model Approximation/Modification (24 Points)

Let $M = (S, A, R, P, \gamma)$ be an MDP with $|S| < \infty$, $|A| < \infty$ and $\gamma \in [0, 1)$. Let $\hat{M}$ be a modification of M to be specified below. We compare the optimal value functions and policies in M and $\hat{M}$.

(a) (12 Points) Let $\hat{M} = (S, A, \hat{R}, P, \gamma)$ where $|\hat{R}(s, a) - R(s, a)| \leq \epsilon$ for all $s \in S$ and $a \in A$. Besides the rewards, all other components of $\hat{M}$ stay the same as in M. Prove that $V^* - \hat{V}^* \leq \frac{\epsilon}{1-\gamma}$. Will M and $\hat{M}$ have the same optimal policy? Briefly explain. Note V^* and $\hat{V}^*$ are optimal value functions in M and $\hat{M}$, respectively.

Solution: Define the state $\tilde{s}$ as follows:

$$\tilde{s} = arg\max_s V^*(s) - \hat{V}^*(s) \tag{1}$$

Also, the inequality $|R(s, a) - \hat{R}(s, a)| \leq \epsilon$ implies:

$$R(s, a) \leq \hat{R}(s, a) + \epsilon \quad \text{for all, } s, a \tag{2}$$

As a result:

$$\begin{aligned} V^*(\tilde{s}) &= \max_a \{R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^*(s')\} \\ &\leq \max_a \{\hat{R}(s, a) + \epsilon + \gamma \sum_{s'} P(s'|s, a)[V^*(s') - \hat{V}^*(s') + \hat{V}^*(s')]\} \\ &\leq \max_a \{\hat{R}(s, a) + \gamma \sum_{s'} P(s'|s, a)\hat{V}^*(s')\} + \max_a \{\gamma \sum_{s'} P(s'|s, a)[V^*(s') - \hat{V}^*(s')]\} + \epsilon \\ &\leq \hat{V}^*(\tilde{s}) + \gamma(V^*(\tilde{s}) - \hat{V}^*(\tilde{s})) + \epsilon \end{aligned} \tag{3}$$

Therefore,

$$V^*(\tilde{s}) \leq \hat{V}(\tilde{s}) + \gamma(V^*(\tilde{s}) - \hat{V}^*(\tilde{s})) + \epsilon$$

Thus,

$$V^*(\tilde{s}) - \hat{V}^*(\tilde{s}) \leq \frac{\epsilon}{1 - \gamma} \tag{4}$$

Hence, for all states s,

$$V^*(s) - \hat{V}^*(s) \leq \frac{\epsilon}{1 - \gamma} \tag{5}$$

$V^*(s)$ and $\hat{V}^*(s)$ do not necessarily have the same optimal policy. This is because the difference in rewards does not necessarily preserve that ordering amongst actions in the state-action values.

(b) (12 Points) Now, let $\hat{M} = (S, A, \hat{R}, P, \gamma)$ where $\hat{R}(s,a) - R(s,a) = \epsilon$ for all $s \in S$ and $a \in A$ for some constant ϵ. Equivalently, the same constant ϵ is added to each reward $R(s,a)$. How are V^* and $\hat{V}^*$ related? Express $\hat{V}^*$ in terms of V^*. Will M and $\hat{M}$ has the same optimal policy? Briefly explain.

Solution: For all s,

$$V^*(s) = \max_a \{R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')\}$$

Since $\frac{\epsilon}{1-\gamma} = \epsilon + \gamma \frac{\epsilon}{1-\gamma}$,

$$V^*(s) + \frac{\epsilon}{1-\gamma} = \max_a \{R(s,a) + \epsilon + \gamma \sum_{s'} P(s'|s,a)(V^*(s') + \frac{\epsilon}{1-\gamma})\}$$
$$\Rightarrow V^*(s) + \frac{\epsilon}{1-\gamma} = \max_a \{\hat{R}(s,a) + \gamma \sum_{s'} P(s'|s,a)(V^*(s') + \frac{\epsilon}{1-\gamma})\}$$

As a result, for all s:

$$\hat{V}^*(s) = V^*(s) + \frac{\epsilon}{1-\gamma}$$

In addition,

$$arg\max_a \{\hat{R}(s,a) + \gamma \sum_{s'} P(s'|s,a) \hat{V}^*(s')\} = arg\max_a \{R(s,a) + \epsilon + \gamma \sum_{s'} P(s'|s,a)(V^*(s') + \frac{\epsilon}{1-\gamma})\}$$
$$arg\max_a \{R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') + \epsilon + \frac{\epsilon}{1-\gamma}\} = arg\max_a \{R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')\}$$

So, M and $\hat{M}$ will have the same optimal policy.

5 Q-Learning (16 Points)

An agent is exploring an MDP $M = (S, A, R, P, \gamma)$ where $S = \{s_1, s_2, s_3\}$ and $A = \{a_1, a_2, a_3\}$ and $\gamma = 0.5$ and $P(s, a_i, s_i) = 1$ for any s for all i. The rewards for transitioning into a state are defined as $R(s_i) = i$. The maximum reward is 3.

(a) (8 Points) The agent follows the trajectory

$$(s1, a1, 1, s1, a2, 2, s2)$$

Consider a Q-learning agent using ϵ-greedy. A random action never happens to pick the greedy action. Ties are broken by choosing the a_i with the smallest i. The learning rate α is set to 0.5. Initialize Q to zeros.

Could such an agent generate this trajectory for $\epsilon \neq 0$? If yes, label the actions that are greedy and which are random, and justify your answers.

Solution: Clearly, it is possible.

For the first experience tuple $(s_1, a_1, 1, s_1)$, we start at s_1 and select action a_1, which returns the reward as 1 and stay at the same state. The initial Q for all state-action is zeros. Since that ties are broken by choosing a_i with the smallest index i, the greedy action is thus a_1. Also, the random action does not pick the greedy action, thus, the random action is a_2 and a_3 here.

For the second experience tuple $(s_1, a_2, 2, s_2)$, we start at s_2 and select action a_2, which returns the reward 2 and goes to state s_2. Now, we've updated the $Q(s_1, a_1)$ based on the first experience tuple, and $Q(s_1, a_1) = 0.5$. Thus the optimal action for s_1 is a_1 now. The action a_2 could be taken only when the action is randomly selected.

(b) (8 Points) Could the `Rmax` algorithm with $m = 1$ have generated this trajectory? Are any of the (s, a) pairs *known* at the end of the trajectory, and if so, which?

Solution:

Yes.

At the beginning, we have

$$R_{\max} = 3$$

, all state actions pairs are unknown, the transitions are initialized to self-loops, and the Q values are initialized to

$$\frac{R_{\max}}{1-\gamma} = \frac{3}{1-0.5} = 6$$

At the first step, as every action has the same Q-value, we break ties and chose action a_1 (could be with the tie breaking from previous question or random). We observe transition $(s_1, a_1, 1, s_1)$.

Then, we update the count for $N(s_1, a_1)$ to 1, and its estimate of $\hat{R}(s_1, a_1) = 1/1 = 1$, and $T(s_1|s_1, a_1) = 1$. As $m = 1$, this state-action tuple becomes known, and we have to update its Q-value

$$\begin{aligned} Q(s_1, a_1) &= \hat{R}(s_1, a_1) + \gamma \sum_{s'} T(s'|s_1, a_1) \max_{a'} Q(s', a') \\ &= 1 + 0.5T(s_1|s_1, a_1)6 \\ &= 4 \end{aligned}$$

At this point, we are in s_1, and

$$\begin{aligned} Q(s_1, a_1) &= 4 \\ Q(s_1, a_2) &= 6 \\ Q(s_1, a_3) &= 6 \end{aligned}$$

Then, the argmax of Q values would be either action a_2 or a_3. We break ties by choosing a_2. We observe $(s_1, a_2, 2, s_2)$. We update the counts

$$\begin{aligned} N(s_1, a_2) &= 1 \\ R(s_1, a_2) &= 2 \end{aligned}$$

And thus, (s_1, a_2) becomes known, and we update its Q value

$$\begin{aligned} Q(s_1, a_2) &= \hat{R}(s_1, a_2) + \gamma \sum_{s'} T(s'|s_1, a_2) \max_{a'} Q(s', a') \\ &= 2 + 0.5T(s_1|s_1, a_1)6 \\ &= 5 \end{aligned}$$

At the end of the trajectory, all state-actions tuples are still unknown, except

$$\begin{aligned} &(s_1, a_1) \\ &(s_1, a_2) \end{aligned}$$

6 Neural Networks (12 Points)

Imagine there is a world where its states are determined by N political parties and represented by a bit string of length N. Each state corresponds to a possible composition of the government with some parties holding seats in the government and others without seats. More specifically, the i-th party holds a seat in a given state if and only if the i-th bit of the corresponding bit string is 1. We say the value of a state is 0 if there are an even number of political parties with a seat in office (because each party is argumentative and can vote for the opposite of each other), or 1 if there are an odd number of political parties with a seat (because ties in voting are not possible). We want to represent the value function by linear function approximation where the input is the binary feature vector s of a state with $s_i = 1$ if i-th party holds a seat, and $s_i = 0$ otherwise.

(a) (3 Points) If $N = 1$, can we represent the value function perfectly using linear value function approximation? Briefly explain.

Solution: Yes.
Let $f(x) = x$. Then the value is 1 when $s = 1$ (1 party in office) and 0 when $s = 0$ (0 parties in office).

(b) (3 Points) If $N = 2$, can we represent the value function perfectly using linear value function approximation? Briefly explain.

Solution: No.
As with the XOR function, this function is not linearly separable. The system of equations

$$\begin{aligned} 0 &\geq w_1(0) + w_2(0) + b = b \\ 0 &< w_1(0) + w_2(1) + b = w_2 + b \\ 0 &< w_1(1) + w_2(0) + b = w_1 + b \\ 0 &\geq w_1(1) + w_2(1) + b = w_1 + w_2 + b \end{aligned}$$

is inconsistent.

(c) (3 Points) Consider using a 2 layer (one hidden layer and 1 output layer) neural network. The hidden neurons are defined as $y = f(w^\intercal x + b)$ for an input x with weight w and bias b and activation function $f(x) = 1$ if $x > 0$ and 0 if $x \leq 0$. Can we represent the value function perfectly for $N = 2$? For any N? Briefly explain your answers.

Solution: Yes, by the universal approximation theorem.
An example of weights that would work for $N = 2$ are $w_{h1} = [1, -1]^\top$, $w_{h2} = [-1, 1]^\top$, and $w_o = [1, 1]^\top$, all with no biases.

(d) (3 Points) For $N = 2$, is there a reason to use a deeper neural network? How about for general N? Briefly justify your answers.

Solution: For $N = 2$, no need because the network is already simple enough.
For general N, deeper layers can help reduce the number of parameters since a single hidden layer would require an exponential number of nodes.