STUDENT NAME: Chenrui Lei

STUDENT ID: 1366324

Practice Exam Questions
CMPUT 366:
Intelligent Systems

University of Alberta
Department of Computing Science

This is (practice for) an in-class closed-book exam. You can mark, write, or sketch your answers directly on the exam. An appropriate amount of blank space is left for answering each question, but feel free to use the backs of pages if necessary. Read each question carefully. **Be sure to justify your answer when asked to do so.**

This examination is closed-book, closed computers, no calculators. You may have exactly one piece of standard-size paper of notes, but they must be your own notes.

Start by writing down your name now. Answer the questions directly on the exam pages in the space provided. Use the backs of the pages if necessary.

The point value of each question and of each of its parts is often indicated in brackets. Partial credit will be given for incomplete or partially correct answers. Read each question carefully and specifically answer each subquestion asked.

Good luck!

1. [1 point]

   Suppose you are using minimax for chess. Your old computer let you search ahead 2 ply. Your new machine is more powerful, so you can search ahead 4 ply. Would the values returned by the minimax function be different between the two machines or not? Justify your answer.

   No, the value returned should stay the same. More powerful machine will get the return value much fast, but the value should be the same since the amount of plys that a machine can search won't affect the accuracy of minmax algorithm.

2. (6 pts total) **True or False and justify your answer**:

   (a) **T** **F** (3 pts) If a policy is greedy with respect to the value function for the equiprobable random policy, then it is an optimal policy.

   (b) **T** **F** (3 pts) If a policy π is greedy with respect to its own value function, v_π, then it is an optimal policy.

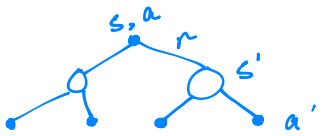
3. All about q_π (22 pts total)

Consider the value function q_π for a continuing problem with discounting.

(a) [3 pts] Give an equation *defining* $q_\pi(s, a)$ in terms of the subsequent rewards $R_{t+1}, R_{t+2}, \ldots$ that would follow if you were in s, a at time t. (If you choose to write it in terms of the return, G_t, define your return notation in terms of the underlying rewards.)

$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ where $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots$

(b) [3 pts] Sketch the backup diagram for q_π. Find a place to attach the labels s, a, r, s' and a'.



(c) [4 pts] What is the Bellman equation for q_π? Write it in an explicit form in terms of $p(s'|s, a)$ and $r(s, a, s')$ so that no expected value notation appears.

$q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q_\pi(s', a')\right], \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$

(d) [4 pts] Consider the simplest, asynchronous dynamic-programming algorithm for computing q_π. An array $q(s,a)$ is initialized to zero. Then there are repeated sweeps through the state–action space, with an update done for each state–action pair. What is that DP update?

$$q(s,a) \leftarrow \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q(s',a')\right], \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

(e) [4 pts] Consider the simplest temporal-difference learning method for estimating q_π from experience. An array $Q(s,a)$ is initialized to zero. Then an infinite sequence of experience, $S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots$, is processed, with one update to $Q(S_t, A_t)$ done for each transition. What is the equation for that TD update?

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$$

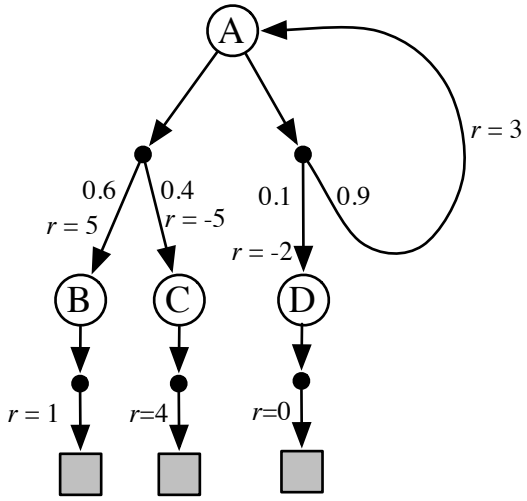
(f) [4 pts] Now consider the simplest Monte Carlo learning method for estimating q_π from episodic experience. An array $Q(s,a)$ is initialized to zero. Then an infinite sequence of episodes is experienced, where an individual episode of experience is denoted $S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots, R_T, S_T$, where T is the final time step of the episode, S_T is the terminal state, and the value of all actions from the terminal state are taken to be zero. When an episode is processed, one update to $Q(S_t, A_t)$ is made for each time step $t < T$. What is that Monte-Carlo update? (If you choose to write it in terms of the return, define your return notation in terms of the underlying rewards.)

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[G_t - Q(S_t, A_t)\right]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

4. MDP values (7 points total)

Consider the following fragment of an MDP graph, where open circles represent states, solid circles represent actions, and boxes represent the terminal state. The fractional numbers indicate the world's transition probabilities and the whole numbers indicate expected rewards. The discount rate is 0.9.



(a) (2 pts) Under the policy that always takes the left-side action out of state A, what are the values of the four states?

$v_\pi(B) = 1$

$v_\pi(C) = 4$

$v_\pi(D) = 0$

$v_\pi(A) = 0.6 \times (5 + 0.9 \times 1) + 0.4 \times (-5 + 0.9 \times 4) = 6.96$

(b) (2 pts) Under the policy that always takes the right-side action out of state A, what are the values of the four states?

$v_\pi(B) = 1$

$v_\pi(C) = 4$

$v_\pi(D) = 0$

$v_\pi(A) = 0.1 \times (-2 + 0.9 \times 0) + 0.9 \times (3 + 0.9 \times v_\pi(A)) = \frac{5}{38}$

(c) (3 pts) Under the policy that takes both actions out of state A with equal probability, what is the value of state A?

$v_\pi(A) = 0.5 \times 6.96 + 0.5 \times [0.1 \times (-2 + 0.9 \times 0) + 0.9 \times (3 + 0.9\, v_\pi(A))]$

$= 3.49 + 0.5 \times [2.5 + 0.81\, v_\pi(A)]$

$= 3.49 + 1.75 + 0.405\, v_\pi(A)$

$v_\pi(A) = \frac{5.24}{0.595} = \frac{1048}{119}$

5. (5 pts) For a finite continuing discounted MDP with discount factor γ, suppose you know two numbers $r_{\min}$ and $r_{\max}$ such that for all immediate rewards R_t, $r_{\min} \leq R_t \leq r_{\max}$. Give epresssions for two numbers $v_{\min}$ and $v_{\max}$ such that $v_{\min} \leq v_\pi(s) \leq v_{\max}$ for all states $s \in S$ and all policies π.

$$v_{min} = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\,[r + \gamma v_{min}]$$

$$v_{max} = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\,[r + \gamma v_{max}]$$

6. (5 pts) What is generalized policy iteration? Refer to all three words of the phrase in your explanation.

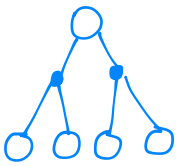
7. Backup diagrams (15 points total)

   A backup diagram is a pattern of nodes and edges representing the update done by a particular algorithm for updating a value function. The node for the state or state-action pair being updated goes at the top and the things it is updated from go below. Open circles are used to represent state nodes, solid circles for action nodes, and squares for terminal states.

   Sketch the backup diagrams for the following tabular algorithms. It is not necessary to attach label to the nodes or links.

   (a) (3 pts) TD(0)

   (b) (3 pts) single-step full backup of v_π



   (c) (3 pts) TD(1)

(d) (3 pts) Two-step Sarsa

(e) (3 pts) Monte Carlo backup for q_*

8. Susan is using reinforcement learning to find the shortest path through a stochastic gridworld with obstacles, a sort of maze. The problem is stochastic in that the four actions, `up`, `down`, `right`, and `left`, cause movement in the corresponding direction with only a slightly higher probability than the other three directions. To encourage the agent to find a short path, Susan formulates the problem as a discounted, episodic MDP, with a reward that is +1 at the terminal goal state and is zero otherwise. She uses ε-greedy Q-learning and off-line updating. Although the agent learns to behave in roughly the right way, Susan is disappointed because the learning agent sometimes finds a policy that is slightly suboptimal with respect to its expected time to the goal, even if α is decreased over time toward zero.

   (a) (4 pts) Why might Susan be having this problem?

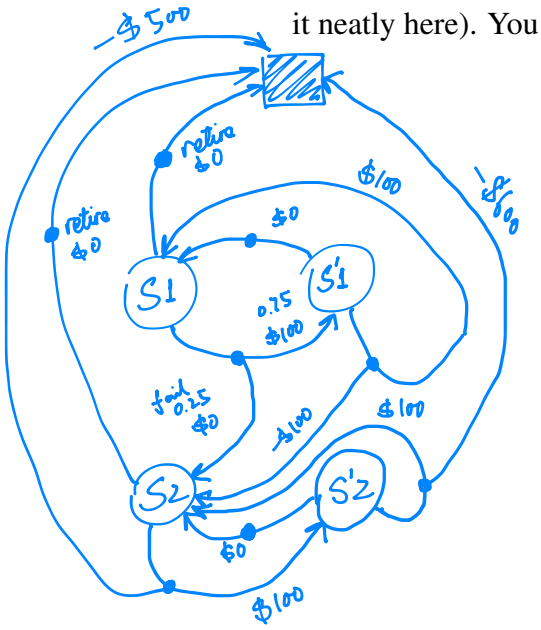
   (b) (2 pts) Can you recommend an alternative *algorithm* that will solve the problem? Why or why not?

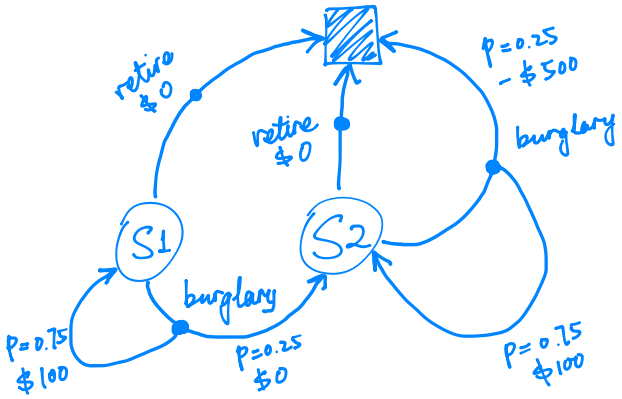
   (c) (2 pts) Is there another change Susan could make so that the agent can be guaranteed to behave as desired?

9. Burglar MDP (12 points)

   At the beginnning of each night a burglar has to decide whether to retire from burgling or, if not, whether to attempt one or two burglaries in that night. Each successful burglary has a value of $100. Each burglary has a probability of .75 of being successful. The first time she is caught she loses all her gains for that night (and cannot commit any further burglaries that night) and is put on probation indefinitely. If she is caught a second time, at any time, she goes to jail and retires from the profession. She values this circumstance as equivalent to a loss of $500.

   (a) (5 pts) Formulate the burglar's problem as a discounted MDP. What are the states, actions, state transition probabilities, and reward function? Suggestion: answer the question by drawing a state-transition diagram and labeling it carefully and clearly. Hint: treat in-jail and retired as the same absorbing state (and work it out on the back of a page, then draw it neatly here). You don't need to express the fractions as decimals.



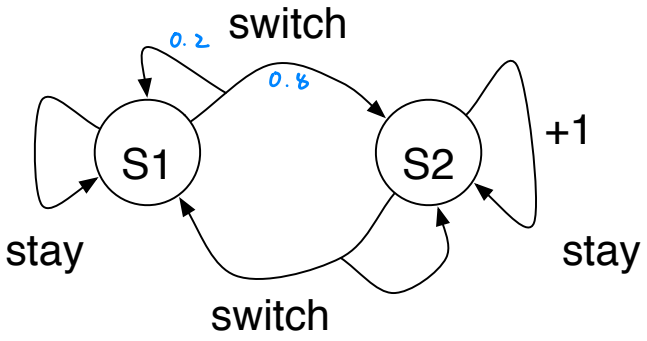


   (b) (3 pts) Explicitly, using numbers or fractions, give the Bellman optimality equations for this MDP.

(c) (4 pts) Assume the burglar has just been caught for the first time while trying to commit the single burglary she had planned for that night. This doesn't stop her, however, from doing a Sarsa(λ) backup. Give a complete description of the operations she would have to do as a result of this experience. Make your description specific to this problem, i.e., don't just give the general Sarsa(λ) equations. Also, don't forget to say how the eligibility traces would be updated.

10. Markov Decision Processes (27 points total)

Consider the MDP in the figure below. There are two states, *S1* and *S2*, and two actions, *switch* and *stay*. The *switch* action takes the agent to the other state with probability 0.8 and stays in the same state with probability 0.2. The *stay* action keeps the agent in the same state with probability 1. The reward for action *stay* in state *S2* is 1. All other rewards are 0. The discount factor is $\gamma = \frac{1}{2}$.



$\frac{a_1(1-q^n)}{1-q}$

(a) [3 points] What is the optimal policy?

S_1 : switch
S_2 : stay

(b) [8 points] Compute the optimal value function by solving the linear system of equations corresponding to the optimal policy.

$v^*(s_2) = 1 + \gamma + \gamma^2 + \cdots = \lim_{n\to\infty} \frac{1-\gamma^n}{1-\gamma} = \frac{1}{1-\gamma} = 2$

$v^*(s_1) = 0.2 \times \frac{1}{2} \times v^*(s_1) + 0.8 \times (1 + \frac{1}{2} \times v^*(s_2))$

$= \frac{1}{10} v^*(s_1) + \frac{8}{5}$

$v^*(s_1) = \frac{8}{5} \times \frac{10}{9} = \frac{16}{9}$

(c) [8 points] Suppose that you are doing synchronous value iteration to compute the optimal state-value function. You start with all value estimates equal to 0. Show the value estimates after 1 and 2 iterations respectively.

1 iteration,

$$v_{\pi^*}(S1) \leftarrow 0.2 \times \frac{1}{2} \times v_{\pi^*}(S1) + 0.8\left(1 + \frac{1}{2} \times v_{\pi^*}(S2)\right)$$
$$\leftarrow 0.8$$
$$v_{\pi^*}(S2) \leftarrow 1 + \gamma \times v_{\pi^*}(S2)$$
$$\leftarrow 1$$

2 iteration,

$$v_{\pi^*}(S1) \leftarrow 0.2 \times \frac{1}{2} \times v_{\pi^*}(S1) + 0.8\left(1 + \frac{1}{2} \times v_{\pi^*}(S2)\right)$$
$$\leftarrow 0.08 + 1.2$$
$$\leftarrow 1.28$$

$$v_{\pi^*}(S2) \leftarrow 1 + \gamma \times v_{\pi^*}(S2)$$
$$\leftarrow 1 + \frac{1}{2} \times 1$$
$$\leftarrow 1.5$$

(d) [8 points] Suppose you are doing TD-learning. You start with all value estimates equal to 0, and you observe the following trajectory (sequence of states, actions and rewards):

S1,switch,0,S2,stay,+1,S2

Assuming the learning rate $\alpha = 0.1$, show the TD-updates that are performed.

$t=1$, $V(S1) \leftarrow V(S1) + \alpha[0 + \gamma V(S2) - V(S1)]$
$\leftarrow 0 + 0.1 \times [0 + \frac{1}{2} \times 0 - 0]$
$\leftarrow 0$

$t=2$, $V(S2) \leftarrow V(S2) + \alpha[1 + \gamma V(S2) - V(S2)]$
$\leftarrow 0 + 0.1 \times [1 + \frac{1}{2} \times 0 - 0]$
$\leftarrow 0.1$

11. (7 pts) What are the key differences between on-policy and off-policy learning? (point form ok)

12. (5 pts) What is the difference between off-policy learning and off-line updating?

13. (5 pts total) Many reinforcement learning algorithms, such as TD(λ), can be viewed in two ways, the forward view and the backward view. The former is particularly useful for theoretical analysis and the latter is particularly useful for developing and understanding mechanisms for online implementation of the algorithms. For each of the following concepts, indicate whether it is primarily associated with the forward view (**F**) or the backward view (**B**):

    F **B** (1 pt) eligibility traces

    F **B** (1 pt) backup diagrams

    F **B** (1 pt) n-step returns

    F **B** (1 pt) the λ-return R_t^λ

    F **B** (1 pt) the TD error δ_t

14. Problem formulation (18 points total)

The *Genius* is a gas-electric hybrid car that automatically switches between two modes, one using an electric motor and one using a gas engine, depending on the battery level and the desired acceleration as indicated by the position of the accelerator foot pedal. When the gas engine is used, the battery is charged at a constant rate up to its maximum capacity. When the electric motor is used, the battery is depleted at a rate proportional to the desired acceleration. Whichever power mode is used, the engine or motor is automatically run so as to achieve the desired acceleration. A decision about which mode to use is made once each second, and we would like to use reinforcement learning to find a decision-making policy that minimizes gas consumption, which is zero when using the electric motor and proportional to the desired acceleration when using the gas engine, in a discounted sense.

This system constitutes a continuous-state MDP. Specify the

(i) (3 pts) states

On eletric motor

On gas engine

(ii) (3 pts) actions

switch

stay

(iii) (3 pts) rewards

the gas comsumption in 1 second

(iv) (3 pts) time steps

1 second

(v) (3 pts) returns

the total gas comsumption to achieve the desired accelerate

(vi) (3 pts) values (in general terms)

for each desired accelerate, there is a value represent the minimal gas comsumption to achieve.