Assignment #2

Course: *Reinforcement Learning (CS6700)*
Instructor: *Prashanth L.A.*
Due date: *April 20th, 2021*

Instructions

1. Work on your own. You can discuss with your classmates on the problems, use books or web. However, the solutions that are submitted must be your own and you must acknowledge all the sources (names of people you worked with, books, webpages etc., including class notes.) Failure to do so will be considered cheating. Identical or similar write-ups will be considered cheating as well.

2. In your submission, add the following declaration at the outset:
"I pledge that I have not copied or given any unauthorized assistance on this assignment."

3. The assignment has two parts. The first part involves theoretical exercises, while the second part requires programming. For the first part, write/typeset the solutions, and upload it on moodle. For the second part, you are required to submit your work in a separate interface (check the details in Section II below).

4. The submission deadline is final, and late submissions would not be considered.

I. Theory exercises

Problem 1.

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a contraction mapping w.r.t. the max-norm. Let α be the modulus, and x^* the fixed point of f. Suppose f satisfies

$$f(x_1) \leq f(x_2), \text{ whenever } x_1 \leq x_2.$$

Assume that there exists a $x' \in \mathbb{R}^n$, and constants c, d such that $ce \leq f(x') - x' \leq de$. Here e denotes the n-vector of ones.
Show that (5 marks)

$$x' - \left(\frac{|c|}{1-\alpha}\right) e \leq f(x') - \left(\frac{|c|\alpha}{1-\alpha}\right) e \leq x^* \leq f(x') + \left(\frac{|d|\alpha}{1-\alpha}\right) e \leq x' + \left(\frac{|d|}{1-\alpha}\right) e.$$

Problem 2.

Consider an n-state discounted problem with bounded single stage cost $g(i,a)$, discount factor $\alpha \in (0,1)$, and transition probabilities $p_{ij}(a)$. For each $j = 1, \ldots, n$, let

$$m_j = \min_{i=1,\ldots,n} \min_{a \in A(i)} p_{ij}(a).$$

For all i, j, a, let

$$\tilde{p}_{ij}(a) = \frac{p_{ij}(a) - m_j}{1 - \sum_{k=1}^n m_k},$$

assuming $\sum_{k=1}^n m_k < 1$.

Answer the following: (1 + 4 marks)

(a) Show that $\tilde{p}_{ij}$ are transition probabilities.

(b) Consider a modified discounted cost problem with single stage cost $g(i,a)$, discount factor $\alpha \left(1 - \sum_{j=1}^n m_j\right)$, and transition probabilities $\tilde{p}_{ij}(a)$. Show that this problem has the same optimal policy as the original, and that its optimal cost $\tilde{J}$ satisfies

$$J^* = \tilde{J} + \frac{\alpha \sum_{j=1}^n m_j \tilde{J}(j)}{1-\alpha} e,$$

where J^* is the optimal cost vector of the original problem, and e is a n-vector of ones.

Problem 3.

Consider a variant of the discounted MDP where in state x_k at time instant k, a coin with bias $\beta \in (0,1)$ is tossed. If the coin turns up heads, then the MDP will transition to the next state according to the underlying transition probabilities. On the other hand, if the coin turns up tails, then the MDP will transition to a special cost-free and absorbing state, say T.

Answer the following: $(3+1$ marks)

1. Show that the variant described above can be cast into the discounted MDP framework.
2. What would be the discount factor of the MDP variant? Would the MDP variant continue to be of discounted type, if the discount factor α of the original MDP is 1?

Problem 4.

Consider an MDP with a finite-horizon. For this problem, derive a policy iteration algorithm. In particular, provide the policy evaluation and policy improvement steps. (4 marks)

Problem 5.

In the "We Care" software firm, every six months, a manager has to decide on "incentives" for his/her managees. Each managee is either of 'Type I' or 'Type II'. The first type managee works for the salary, while the second type managee works out of passion. The manager can choose to give perks to all the managees of a single type, and has to decide which managee class to incentivize. The cost of the incentive is 5000 INR per managee. If a managee of 'Type I' gets an incentive, then he/she contributes 7500 INR to company's profit, while his/her contribution in case of not receiving an incentive is 2500 INR. The corresponding numbers for Type-II managees are 20000 INR and 10000 INR, respectively. Note that the type of a managee is not static, and incentives are doled out at the beginning of a six-month period. If a Type-I managee receives an incentive, then with probability (w.p) 0.75 he/she becomes a "Type-II" person at the end of a six-month period, and in case he/she does not receive an incentive, w.p. 0.75 he/she stays as Type-I. A Type-II person w.p. 0.8 stays Type-II on receiving an incentive, and w.p. 0.4 stays Type-II if not given an incentive.

Answer the following: (3+2+2 marks)

(a) Formulate this as an infinite-horizon discounted MDP, with discount factor 0.9.

(b) Find an optimal policy using policy iteration starting with a greedy policy that picks an action with the highest single-stage reward.

(c) Run value iteration, and Gauss- Seidel value iteration with $\delta = 0.1$.

For the last two parts, you may use a computer program. In any case, tabulate/plot the iterate value for each iteration.

II. Simulation exercises

Question 1. Consider a problem of a taxi driver, who serves three cities A, B and C. The taxi driver can find a new ride by choosing one of the following actions.

1. Cruise the streets looking for a passenger.
2. Go to the nearest taxi stand and wait in line.
3. Wait for a call from the dispatcher (this is not possible in town B because of poor reception).

For a given town and a given action, there is a probability that the next trip will go to each of the towns A, B and C and a corresponding reward in monetary units associated with each such trip. This reward represents the income from the trip after all necessary expenses have been deducted. Please refer Table 1 below for the rewards and transition probabilities. In Table 1 below, p_{ij}^k is the probability of getting a ride to town j, by choosing an action k while the driver was in town i and r_{ij}^k is the immediate reward of getting a ride to town j, by choosing an action k while the driver was in town i.

Town i	**Actions** k	**Probabilities** p_{ij}^k j = A B C	**Rewards** r_{ij}^k j = A B C
A	1 2 3	A B C $\begin{bmatrix} 1/2 & 1/4 & 1/4 \\ 1/16 & 3/4 & 3/16 \\ 1/4 & 1/8 & 5/8 \end{bmatrix}$	A B C $\begin{bmatrix} 10 & 4 & 8 \\ 8 & 2 & 4 \\ 4 & 6 & 4 \end{bmatrix}$
B	1 2	A B C $\begin{bmatrix} 1/2 & 0 & 1/2 \\ 1/16 & 7/8 & 1/16 \end{bmatrix}$	A B C $\begin{bmatrix} 14 & 0 & 18 \\ 8 & 16 & 8 \end{bmatrix}$
C	1 2 3	A B C $\begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/8 & 3/4 & 1/8 \\ 3/4 & 1/16 & 3/16 \end{bmatrix}$	A B C $\begin{bmatrix} 10 & 2 & 8 \\ 6 & 4 & 2 \\ 4 & 0 & 8 \end{bmatrix}$

Table 1: Taxi Problem: Probabilities and Rewards

Suppose $1-\gamma$ is the probability that the taxi will breakdown before the next trip. The driver's goal is to maximize the total reward untill his taxi breakdown.

Implement the following. $(1.5+1+1.5+1.5+1.5$ marks$)$

1.1: Find an optimal policy using **policy iteration**(Algorithm 3) starting with a policy that will always cruise independent of the town, and a zero value vector. Let $\gamma = 0.9$.

1.2: Run **policy iteration** for discount factors γ ranging from 0 to 0.95 with intervals of 0.05 and display the results.

1.3: Find an optimal policy using **modified policy iteration**(Algorithm 4) starting with a policy that will always cruise independent of the town, and a zero value vector. Let $\gamma = 0.9$ and $m = 5$.

1.4: Find optimal values using **value iteration**(Algorithm 1) starting with a zero vector. Let $\gamma = 0.9$.

1.5: Find optimal values using **Gauss-Seidel value iteration**(Algorithm 2) starting with a zero vector. Let $\gamma = 0.9$.

Answer the following questions. $(1 + 1 + 1$ marks)

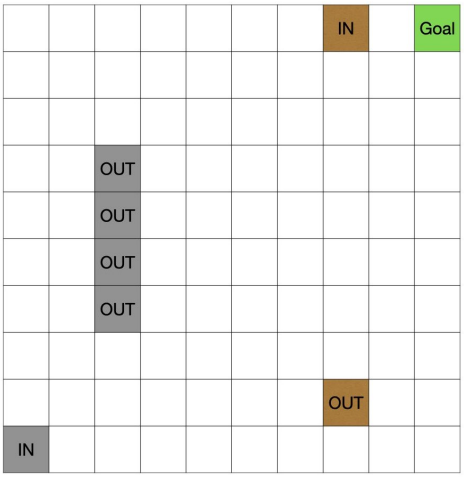
1.a How is different values of γ affecting the **policy iteration** from **1.2**? Explain your findings.

1.b For **modified policy iteration** from **1.3**, do you find any improvement if you choose $m = 10$? Explain your findings.

1.c Compare and contrast the behavior of **value iteration** from **1.4** and **Gauss-Seidel value iteration** from **1.5**.

Question 2. Consider a 10×10 gridworld (GridWorld-1) as shown in Figure 1.

Figure 1: GridWorld-1



- **State space**: Gridworld has 100 distinct states. There is a special absorbing state called "Goal". There are two wormholes labeled as "IN" in Grey and Brown, any action taken in those states will teleport you to state labeled "OUT" in Grey and Brown respectively. In case of Grey wormhole you can teleport to any one of the states labeled OUT with equal probability (i.e. $1/4$). States labeled OUT is just a normal state.
- **Actions:** In each non-terminal state, you can take 4 actions $\mathcal{A} = \{$North, East, South, West$\}$, which moves you one cell in the corresponding direction.
- **Transition model:** Gridworld is stochastic because the actions can be unreliable. In this model, an action $X \in \{$North, East, South, West$\}$ moves you one cell in the X direction of your current position with probability 0.8, but with probabilities $0.2/3$ it will transition

to one of the other neighbouring cells. For example, if the selected action is North, it will transition you one cell to the North of your current position with probability 0.8, one cell to the East, West, or South of your current position with probability $0.2/3$.
Transitions that take you off the grid will not result in any change. There are no transitions available from the “Goal" state.

- **Rewards:** You will receive a reward of 0 for all transitions except the transition to the “Goal" state. Any transition to the "Goal" state gives you a reward of $+10$.

Implement the following. $(2+2+2+1$ marks)

2.1: Find optimal values using **value iteration** (Algorithm 1)). Start with $J_0(s) = 0$ and $\pi_0(s) =$ North, $\forall s$. Let the discount factor $\gamma = 0.7$.

2.2: Find optimal policy for each state using **policy iteration** (Algorithm 3). Start with $J_0(s) = 0$ and $\pi_0(s) =$ North, $\forall s$. Let the discount factor $\gamma = 0.7$.

2.3: Implement **TD(λ)** (Algorithm 5). Choose the “Grey IN " state as the start state. Let the discount factor $\gamma = 0.7$, and step-size $\alpha = 0.5$. Let the maximum number of episodes $N = 1000$. Consider the policy π as given in Algorithm 6.

2.4: Run TD(λ) for different values of λ ranging from 0 to 1 with intervals of 0.05 and display the results.

Answer the following questions. $(2+2+2+2$ marks)

2.a Compare **value iteration** from **2.1** and **policy iteration** from **2.2** by plotting $J(s)$ vs iterations (the outer loop iterations) for states “Brown IN", “Brown OUT", and “Grey IN". Which algorithm converges faster? Why?

2.b How is different values of λ affecting the TD(λ) from **2.4**? Explain your findings.

Let $\forall s \in \mathcal{S}$, $J_v(s)$ be the optimal values (final values) obtained using **value iteration** from **2.1**. Then the root mean squared (RMS) error of J with respect to J_v averaged over states can be calulated as

$$error(J) = \sqrt{\frac{\sum_{s \in \mathcal{S}} \left(J(s) - J_v(s)\right)^2}{|\mathcal{S}|}} \tag{1}$$

2.c Plot graph of $error(J^i)$ from **policy iteration** from **2.2** vs iterations i. Here J^i is the values obtained after i^{th} iteration of the outer loop of the algorithm.

2.d Plot graph of $error(J^i)$ from **TD(λ)** from **2.3** vs iterations i for $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$. Here J^i is the values obtained after i^{th} iteration of the outer loop the algorithm.

The pseudocode for the Algorithms are given below. (courtesy: Sutton & Barto 1998)

Algorithm 1 Value Iteration

Initialize: $J(s) = 0, \forall s \in \mathcal{S}$;
repeat
  $\delta = 0$;
  for each $s \in \mathcal{S}$ **do**
    $H(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s,a,s') + \gamma J(s')]$
    $\delta = \max(\delta, |J(s) - H(s)|)$;
  end for
  for each $s \in \mathcal{S}$ **do**
    $J(s) = H(s)$;
  end for
until $(\delta < 1\text{e}{-8})$

Algorithm 2 Gauss-Seidel Value Iteration

Initialize: $J(s) = 0, \forall s \in \mathcal{S}$;
repeat
  $\delta = 0$;
  for each $s \in \mathcal{S}$ **do**
    $j = J(s)$;
    $J(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s,a,s') + \gamma J(s')]$
    $\delta = \max(\delta, |j - J(s)|)$;
  end for
until $(\delta < 1\text{e}{-8})$

Algorithm 3 Policy Iteration

Input: $\pi_0(s), \forall s \in \mathcal{S}$;
Initialize: $J(s) = 0, \pi(s) = \pi_0(s), \forall s \in \mathcal{S}$;
repeat
  repeat
    $\delta = 0$;
    for each $s \in \mathcal{S}$ **do**
      $j = J(s)$;
      $J(s) = \sum_{s' \in \mathcal{S}} P_{ss'}(\pi(s))[r(s,\pi(s),s') + \gamma J(s')]$
      $\delta = \max(\delta, |j - J(s)|)$;
    end for
  until $(\delta < 1\text{e}{-8})$
  $done = 1$;
  for each $s \in \mathcal{S}$ **do**
    $b = \pi(s)$;
    $\pi(s) = \text{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s,a,s') + \gamma J(s')]$;
    if $b \neq \pi(s)$ **then**
      $done = 0$;
    end if
  end for
until $done = 1$

Algorithm 4 Modified Policy Iteration

Input: $\pi_0(s), \forall s \in \mathcal{S}$, m;
Initialize: $J(s) = 0, \pi(s) = \pi_0(s), \forall s \in \mathcal{S}$;
repeat
  for $k = 0, \cdots, m$ **do**
    for each $s \in \mathcal{S}$ **do**
      $J(s) = \sum_{s' \in \mathcal{S}} P_{ss'}(\pi(s))[r(s, \pi(s), s') + \gamma J(s')]$
    end for
  end for
  $done = 1$;
  for each $s \in \mathcal{S}$ **do**
    $b = \pi(s)$;
    $\pi(s) = \text{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s, a, s') + \gamma J(s')]$;
    if $b \neq \pi(s)$ **then**
      $done = 0$;
    end if
  end for
until $done = 1$

Algorithm 5 TD(λ)

Input: Start state s_0, the policy $\pi(s), \forall s \in \mathcal{S}$, maximum number of episodes N, step-size α, and discount factor γ;
Initialize: $J(s) = e(s) = 0, \forall s \in \mathcal{S}$;
for $i = 1, \cdots, N$ **do**
  $s = s_0$;
  repeat {each step of episode}
    $a = \pi(s)$;
    Take action a, observe reward r, and next state s'.
    $\delta = r + \gamma J(s') - J(s)$;
    $e(s) = e(s) + 1$;
    for each $s \in \mathcal{S}$ **do**
      $J(s) = J(s) + \alpha \delta e(s)$;
      $e(s) = \gamma \lambda e(s)$;
    end for
    $s = s'$;
  until s is terminal
end for

Algorithm 6 policy for TD(λ)

Input: Current state position (s_x, s_y), Goal state position (g_x, g_y);
Output: The direction of the action $d \in \{(1,0), (-1,0), (0,1), (0,-1)\}$;
$\delta_x = g_x - s_x$; $\delta_y = g_y - s_y$;
if $|\delta_x| \geq |\delta_y|$ **then**
  $d = (sign(\delta_x), 0)$;
else
  $d = (0, sign(\delta_y))$;
end if
return d;