AI 533 **Intelligent Agents and Decision Making** Winter 2025

OREGON STATE UNIVERSITY

School of Electrical Engineering and Computer Science

Instructor: Sandhya Saisubramanian

Total points: 100 Assignment 1: MDP Due date: Jan 27, 2025

Instructions: Collaboration is not allowed on any part of this assignment. It is acceptable to discuss concepts or clarify questions but you must document who you worked with for this assignment. Copying answers or reusing solutions from individuals/the internet is unacceptable and will be dealt with strictly. Solutions must be typed (hand written and scanned submissions will not be accepted) and saved as a .pdf file.

1. **(15 points)** MDP design. For the following example scenarios, you must (1) write a factored state representation, discuss if it is discrete or continuous, and whether your state representation satisfies the Markov property; (2) *list* the actions (e.g. move up, move right, etc.) the agent should have to accomplish the assigned task; (3) write a reward function such that optimizing the reward function will help the agent accomplish its task efficiently (optimally) and briefly explain your answer. You are free to choose $R(s)$, $R(s,a)$ or $R(s,a,s')$ for your reward function. You can write your reward function mathematically or describe it clearly in text. Proposed reward functions must be reasonable, practical, and must not lead to unsafe behavior.

   (i) A robotic vacuum cleaner is assigned the task of removing dirt from the floor

   The robotic vacuum cleaner's state can be written as $S = \{x, y, D\}$, where x and y are the robot's current position on the grid, and $D(x, y)$ is a binary variable indicating whether the current cell is dirty 1 or clean 0.
   This state representation is discrete because the position and dirt status can take only finite values, and it satisfies the Markov property because the current state provides all necessary information for the robot to decide its next action without requiring any knowledge of past states.
   The robot can perform the following actions: move up, move down, move left, move right, and vacuum (clean the current cell).
   The reward function, $R(s, a)$, is when the robot receives a reward of $+10$ for cleaning a dirty cell gets a penalty of -1 for each movement and gets 0 for vacuuming an already clean cell, and gets a large penalty of -100 for colliding with obstacles such as walls or furniture. This reward function makes sure the robot prioritizes cleaning dirt.

   (ii) A legged robot wants to run a marathon and reach the finish line as quickly as possible

   The state of the legged robot can be written as $S = \{x, v, e\}$, where x is the robot's position along the marathon route, v is its velocity, and e is the remaining energy level of the robot.
   This state representation is continuous because the position and velocity are real-valued variables, and it satisfies the Markov property since the current position, velocity, and energy level provide all necessary information to tell the next state without tracking the history of prior states.
   The robot can perform the following actions: increase velocity, decrease velocity, maintain current velocity, or stop for a recharge.

The reward function, $R(s, a)$, is when the robot gets a high positive reward, $+100$, for crossing the finish line; gets a small penalty, -1, for efficiency; and gets a large penalty, -50, if it depletes its energy completely without reaching the finish line. This reward function pushes the robot to reach the finish line quickly.

(iii) An autonomous car whose decisions must minimize the mean commute for all drivers (those driven by humans and those driven by AI)
The autonomous car's state can be wriiten as $S = \{x, v, t, d_h\}$, where x is the current position of the car on the road network, v is the car's velocity, t is the time elapsed in the commute, and d_h is the density of human-driven vehicles in the area.
This state representation is continuous because position, velocity, and density are real-valued variables, and it satisfies the Markov property because the current state provides all necessary information to decide the next action without requiring the history of prior states.
The car can perform the following actions: accelerate, decelerate, maintain speed, or change lanes.
The reward function, $R(s, a)$, can be designed as $+10$ for reducing commute time for both the autonomous car and nearby human drivers, -1 for actions that increase congestion or lead to inefficiencies, and -100 for unsafe actions that may cause accidents or violations.

(iv) An underwater robot that must monitor the health of corals
The underwater robot's state can be represented as $S = \{x, y, d, h, b\}$, where x, y are the robot's current position in the monitoring area, d is the depth of the robot, h is the health status of the corals in the current location (e.g., normal, stressed, bleached), and b is the battery level of the robot.
This state representation is continuous because position, depth, and battery levels are real-valued variables, and it satisfies the Markov property because the current state contains all necessary information for deciding the next action without needing past states.
The robot can take the following actions: move forward, move left, move right, ascend, descend, or analyze coral health.
The reward function, $R(s, a)$, is designed as follows: $+10$ for successfully collecting data on coral health, -1 for energy consumption with each movement for efficiency, and -100 if the battery runs out before completing the mission.

(v) An autonomous robot that is tasked with irrigating and fertilizing the crops to maximize crop yield, without adversely affecting crop and soil health.
The state of the farming robot can be written as $S = \{x, y, m, s, w, t\}$, where x, y is the robot's position in the field, m is the moisture level of the soil, s is the soil nutrient content, w is the weather condition (e.g., temperature, humidity), and t is the time elapsed since the last irrigation or fertilization.
This state is continuous because moisture levels, soil nutrients, and weather conditions are real-valued variables. It satisfies the Markov property.
The robot can perform the following actions: irrigate, fertilize, move to another location, or remain idle.
The reward function, $R(s, a)$, is defined as follows: $+10$ for actions that improve crop yield without overwatering or over-fertilizing, -1 for unnecessary actions that waste resources, and -100 for actions that cause excessive irrigation or fertilization, leading to soil degradation.

2. **(15 points)** Given an MDP $M = (S, A, T, R, \gamma)$ with a fixed state state s_0 and a fixed policy π, the probability that the action at time $t = 0$ is $a \in A$ is:

$$\Pr(A_0 = a) = \pi(s_0, a).$$

Similarly, the probability that the state at time $t = 1$ is $s \in S$ is:

$$\Pr(S_1 = s) = \sum_{a_0 \in A} \pi(s_0, a_0) T(s_0, a_0, s).$$

Write a similar mathematical expression (using only S, A, T, R, γ, π and Bayes' theorem) for the following:

(i) The expected reward at time $t = 6$ given that the action at time $t = 3$ is $a \in A$ and the state at time $t = 5$ is $s \in S$. Use $R(s, a)$ for reward notation.

$$\mathbb{E}[R_6 \mid A_3 = a, S_5 = s] = \sum_{a_5 \in A} \pi(s, a_5) \sum_{s_6 \in S} T(s, a_5, s_6) \sum_{a_6 \in A} \pi(s_6, a_6) R(s_6, a_6).$$

- $\pi(s, a)$ is the policy, which defines the probability of selecting action
- $T(s, a, s')$ is the transition function
- $R(s, a)$ is the reward function

(ii) The probability that the action at time $t = 16$ is $a' \in A$ given that the action at time $t = 15$ is $a \in A$ and the state at time $t = 14$ is $s \in S$.

The probability that the action can be computed using Bayes' theorem:

$$P(A_{16} = a' \mid A_{15} = a, S_{14} = s) = \frac{P(A_{16} = a', A_{15} = a \mid S_{14} = s)}{P(A_{15} = a \mid S_{14} = s)}.$$

Expanding the joint probability we get:

$$P(A_{16} = a' \mid A_{15} = a, S_{14} = s) = \sum_{s_{15} \in S} \sum_{s_{16} \in S} \pi(s_{16}, A_{16} = a') T(s_{15}, A_{15} = a, s_{16}) T(s, a_{14}, s_{15}).$$

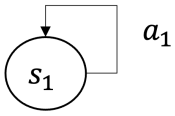
3. **(5 points)** How many deterministic policies (optimal or otherwise) exist for an MDP with 5 states and 10 actions?
A deterministic policy in a MDP is a function that maps each state to a single action.
For an MDP with S states and A actions, the deterministic policy is

$$\pi : S \to A$$

The MDP has S = 5 states and A = 10 actions per state, and the total number of deterministic policies is

$$A^S = 10^5.$$

For an MDP with 5 states and 10 actions, there exists **100,000** deterministic policies (including both optimal and non-optimal policies).



4. **(10 points)** For the MDP in the following figure with one state and one action, let $R(s_1) = 0$ and $V_0(s_1) = 5$.

   (i) Will value iteration converge when $\gamma = 1$? Briefly explain your answer.

   To know whether value iteration will converge under two scenarios for the given MDP for state s_1, action a_1, reward $R(s_1) = 0$, initial value $V_0(s_1) = 5$.
   The state s_1 transitions back to itself with a_1.

   The value iteration update rule for the single state s_1 is:

   $$V_{k+1}(s_1) = R(s_1) + \gamma V_k(s_1)$$

   Here $R(s_1) = 0$

   $$V_{k+1}(s_1) = \gamma V_k(s_1)$$

   Substituting $\gamma = 1$:

   $$V_{k+1}(s_1) = V_k(s_1)$$

   If $V_0(s_1) = 5$, the value at each iteration remains constant:

   $$V_1(s_1) = 1 \cdot 5 = 5, \quad V_2(s_1) = 5, \quad V_3(s_1) = 5, \ldots$$

   The values will not converge to a finite solution as there is no decay in future rewards when $\gamma = 1$. The values remain constant depending on the initial condition so we can say that value iteration does not converge when $\gamma = 1$.

   (ii) Will value iteration converge when $\gamma = 0.9$? Briefly explain your answer.
   Yes, value iteration converges when $\gamma = 0.9$

   For $\gamma = 0.9$, the equation becomes:

   $$V_{k+1}(s_1) = 0.9 V_k(s_1)$$

   Starting with $V_0(s_1) = 5$:

   $$V_1(s_1) = 0.9 \cdot 5 = 4.5, \quad V_2(s_1) = 0.9 \cdot 4.5 = 4.05, \quad V_3(s_1) = 0.9 \cdot 4.05 = 3.66, \ldots$$

   The values decreases as we approach $V^*(s_1) = 0$ over finite iterations.

   The values converge when $\gamma < 1$ makes sure that future rewards diminish geometrically, leading to convergence.

5. **(15 points)** Prove the following two statements mathematically or provide an example to demonstrate the property.

Statement 1: Multiplying all rewards (of a finite, discrete MDP with bounded rewards) by a positive scalar does not change the optimal policy.

To prove that multiplying all rewards in a finite, discrete MDP by a positive scalar does not change the optimal policy. Considering the Bellman optimality equation.
The equation for the optimal value function $V^*(s)$ is:

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

Here $R(s,a)$ is the immediate reward for taking action a in state s, γ is the discount factor ($0 \leq \gamma < 1$), $T(s,a,s')$ is the transition probability to state s' given state s and action a, and $V^*(s)$ is the optimal value of state s.

Multiplying reward by positive scalar $\alpha > 0$

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

Assuming the new value function is scaled by α, so $V'^*(s) = \alpha V^*(s)$.

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

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

This is the original Bellman equation for $V^*(s)$. Thus, $V'^*(s) = \alpha V^*(s)$, that means the value function is scaled by α, but the relative ordering of actions remains unchanged. Since the optimal policy is determined by the action that maximizes the value function, scaling the rewards by a positive scalar $\alpha > 0$ does not change the optimal policy.

Statement 2: Adding a positive constant to all rewards (all states or state-action pairs or state-action-successor pairs in the MDP) of a finite MDP with bounded rewards changes the optimal policy.
The Bellman optimality equation for the value function is:

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

Adding a constant (c¿0) to rewards:

$$R'(s,a) = R(s,a) + c$$

$$V'^*(s) = \max_{a \in A} \left[R(s,a) + c + \gamma \sum_{s' \in S} T(s,a,s') V'^*(s') \right]$$

$$V'^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V'^*(s') \right] + \frac{c}{1-\gamma}.$$

$\frac{c}{1-\gamma}$ is constant across all states and actions

So adding this constant does not affect the action that maximizes the value function. The relative ranking of actions remains unchanged.

The optimal policy depends only on the relative ranking of actions, which is unaffected by a uniform additive constant.

The statement is false. Adding a positive constant to all rewards in an MDP does not change the optimal policy. The only differences in rewards influence the policy and additive constants are canceled out during maximization.

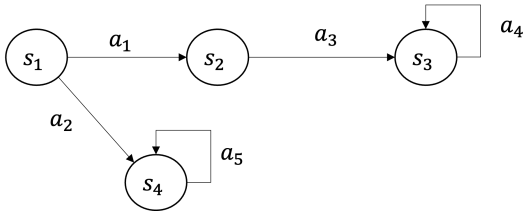
6. **(30 points)** In this question, the goal is to demonstrate that value improvement in value iteration is not monotonic. While the values and the policy converge to optimal at termination, for a finite, bounded MDP, the values (and the resulting policy) before convergence of value iteration are not guaranteed to improve in a monotonic manner.

   Consider the MDP in Figure 1 with four states, five actions that have deterministic transitions, and discount factor $\gamma = 1$. The reward for being in each state, $R(s)$, is reported in the table in Figure 1. Let V_i and V_{i+1} denote value functions from two iterations of value iteration on this problem **before convergence**. Let π_i and π_{i+1} denote the policies that are greedy with respect to these value functions.

   (i) You are required to assign an initial value (V_0) to each of the state such that the expected reward following the greedy policy is not monotonic, $\pi_{i+1} < \pi_i$. Initialize V_0 such that at some finite iteration count i before convergence of VI, the expected reward following π_{i+1} is less than the expected reward following π_i, causing a change in the *current* optimal action at least in one state.

   For example, let a_1 and a_2 denote the actions available in state s. Let the true optimal action in s be $a^* = a_1$. You must initialize V_0 such that at iteration i, the policy that is greedy on the values $\pi(s) = a_1$ but at iteration $i+1$, $\pi(s) = a_2$ and in some $i+k$, the policy stabilizes to the true optimal policy $\pi(s) = a_1$.

   (ii) Show your calculation of value iteration to support (i).



State	Reward R(s)
s_1	0
s_2	-1
s_3	+2
s_4	0

Figure 1: Graph for analyzing monotonicity of value iteration

(i) Initialization

The reward structure is given as:

$$R(s_1) = 0, \quad R(s_2) = -1, \quad R(s_3) = +2, \quad R(s_4) = 0.$$

The Markov Decision Process (MDP) consists of:

- **State Space:** $S = \{s_1, s_2, s_3, s_4\}$
- **Action Space:** $A = \{a_1, a_2, a_3, a_4, a_5\}$
- **Discount Factor:** $\gamma = 1$

State	Reward $R(s)$
s_1	0
s_2	-1
s_3	+2
s_4	0

Table 1: Reward values for each state in the MDP

The Bellman update rule is:

$$V_{k+1}(s) = \max_a \left[R(s) + \sum_{s'} T(s, a, s') V_k(s') \right].$$

Initial setup to induce the policy change, this forces a temporary switch in the policy before final convergence.

$$V_0(s_1) = 0, \quad V_0(s_2) = 0, \quad V_0(s_3) = 2, \quad V_0(s_4) = 0.$$

(ii) Value iterations:

$$V_1(s_1) = \max\{0 + 0, 0 + 0\} = 0$$
$$V_1(s_2) = -1 + 2 = 1$$
$$V_1(s_3) = 2 + 2 = 4$$
$$V_1(s_4) = \max\{0 + 0, 0 + 2\} = 2$$
$$\pi_1 = \{a_1, a_1, a_3, a_5\}$$

State	Value $V_1(s)$	Greedy Policy $\pi_1(s)$
s_1	0	a_1
s_2	1	a_1
s_3	4	a_3
s_4	2	a_5

Table 2: First iteration value function and greedy policy

$$V_2(s_1) = \max\{0 + 1, 0 + 2\} = 2$$
$$V_2(s_2) = -1 + 4 = 3$$
$$V_2(s_3) = 2 + 4 = 6$$
$$V_2(s_4) = \max\{0 + 0, 0 + 4\} = 4$$
$$\pi_2 = \{a_1, a_2, a_3, a_5\}$$

State	Value $V_2(s)$	Greedy Policy $\pi_2(s)$
s_1	2	a_1
s_2	3	a_2
s_3	6	a_3
s_4	4	a_5

Table 3: 2nd iteration value function and greedy policy

The greedy policy might switch actions like from a_1 to a_2 temporarily in some states (e.g., s_2) before settling. During this, the values don't always rise everywhere every step. This shows that value iteration isn't guaranteed to improve values monotonically at every stage before it finally converges, proving non-monotonicity in the process.

Value iteration can behave non-monotonically before converging.

7. **(10 points)** In class, we proved the contraction mapping for the Bellman equation, independent of the policy. You are required to prove that the Bellman backup operator for a particular policy converges. For a deterministic policy π and $0 \leq \gamma < 1$, let us define a contraction operator $(B^\pi V)(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s')V(s')$. Prove that $\|B^\pi V B^\pi V'\| \leq \gamma \|VV'\|$. Hint: use the max-norm operator $\|v\| = max_s|v(s)|$ and follow steps similar to the proof in lecture slides.

For a deterministic policy π, the Bellman operator is:

$$B^\pi V(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s')V(s').$$

To prove

$$\|B^\pi V - B^\pi V'\| \leq \gamma \|V - V'\|, \text{ where the max-norm is:}$$

$$\|v\| = \max_s |v(s)|.$$

$$B^\pi V(s) - B^\pi V'(s) = \gamma \sum_{s'} T(s, \pi(s), s')\big(V(s') - V'(s')\big).$$

$$\|B^\pi V - B^\pi V'\| = \max_s \Big|\gamma \sum_{s'} T(s, \pi(s), s')\big(V(s') - V'(s')\big)\Big|.$$

Since $\sum_{s'} T(s, \pi(s), s') = 1$, we can write the above equation as

$$\|B^\pi V - B^\pi V'\| \leq \gamma \max_s \sum_{s'} T(s, \pi(s), s')|V(s') - V'(s')|.$$

By the convex combination property:

$$\sum_{s'} T(s, \pi(s), s')|V(s') - V'(s')| \leq \|V - V'\|.$$

$$\|B^\pi V - B^\pi V'\| \leq \gamma \|V - V'\|.$$

Since $0 \leq \gamma < 1$, the Bellman operator is a contraction under the max-norm, so B^π converges to a unique fixed point V^π.

$$\boxed{\|B^\pi V - B^\pi V'\| \leq \gamma \|V - V'\|.}$$