CS461 Artificial Intelligence

Fall 2024: Homework 4

Due Date: 10.12.2024

1 Informed Search

Consider a graph where the edge costs are always positive. We want to find the least cost path from a node S to another node G. Suppose we use A* tree search with some heuristic h' to find such a path. Let C denote the cost of the path from S to G by using heuristic h', $h^*(X)$ denote the shortest distance to G from a node X, and h be an admissible heuristic for this graph.

(a) For the given definitions of h' below, determine whether $C = h^*(S)$, $C \geq h^*(S)$, or $C > h^*(S)$.

(i) $h'(X) = h(X)/2$

(ii) $h'(X) = \frac{h(X)+h*(X)}{2}$

(iii) $h'(X) = h(X) + h*(X)$

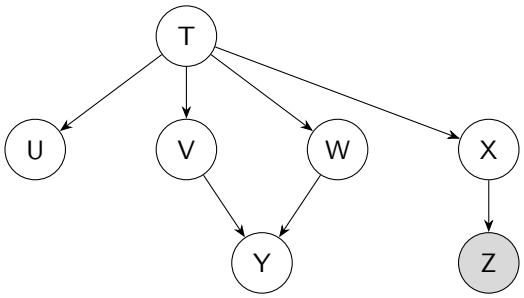
(iv) $h'(X) \leq \begin{cases} \min_{Y \in K(X)} h'(Y) - h(Y) + h(X) & \text{if } K(X) \neq \emptyset, \\ h(X) & \text{if } K(X) = \emptyset. \end{cases}$

where $K(X)$ defines the set of all neighboring nodes Y that satisfy $h^*(X) > h^*(Y)$.

(v) $h'(X) = \begin{cases} \min_{Y \in K(X)} h(Y) + cost(X, Y) & \text{if } K(X) \neq \emptyset, \\ h(X) & \text{if } K(X) = \emptyset. \end{cases}$

2 Bayes Networks - Variable Elimination

Consider the Bayes network below where each variable has a binary domain. We want to find P(Y — +z). In order to find it, we will perform variable elimination with the order: X, T, U, V, W.



1. What factors do you have after inserting the evidence?

2. Find the factor f_1 (as an equation) generated by eliminating X and give the resulting list of factors.

3. Find the factor f_2 (as an equation) generated by eliminating T and give the resulting list of factors.

4. Find the factor f_3 (as an equation) generated by eliminating U and give the resulting list of factors.

5. Find the factor f_4 (as an equation) generated by eliminating V and give the resulting list of factors.

6. Find the factor f_5 (as an equation) generated by eliminating W and give the resulting list of factors.

7. How will you obtain P(Y — +z) from the remaining factors?

8. Is the variable elimination order optimal? If it is not, give an optimal order.

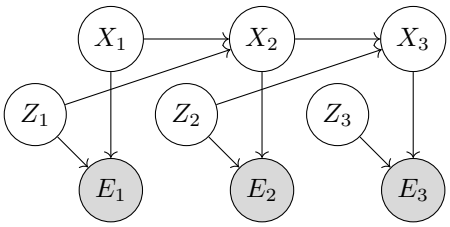
3 HMM - Filtering

Consider the two HMM-like models below. Find the elapse time ($P(X_t, Z_t|e_{1:t-1})$) and observation ($P(X_t, Z_t|e_{1:t})$) updates.

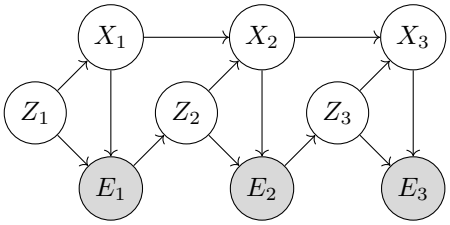
Hint: Recall that for standard HMMs, they are in the following forms, respectively:

$$P(X_t|e_{1:t-1}) = \sum_{x_{t-1}} P(X_t|x_{t-1})P(x_{t-1}|e_{1:t-1})$$

$$P(X_t|e_{1:t}) \propto P(X_t|e_{1:t-1})P(e_t|X_t)$$



(a)



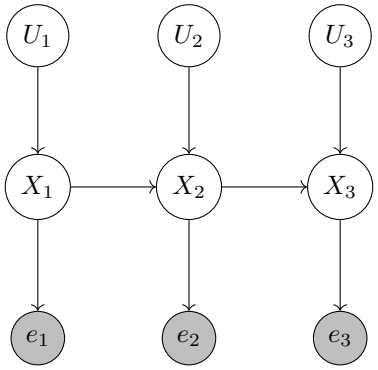
(b)

(a) Equations for model (a)

(b) Equations for model (b)

4 HMM: Viterbi Algorithm

Consider the HMM structure below. The agent performs an action U_t from some state-independent policy at every time step t, and also receives some evidence e_t of the true state X_t.



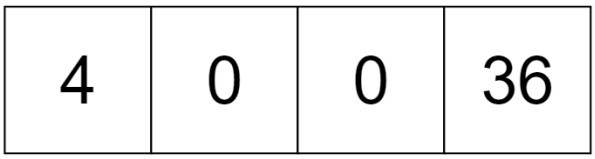
In this setting, which (if any) of the expressions below are maximized by the Viterbi algorithm? Explain your reasoning below.
Hint: Remember that the Viterbi algorithm for vanilla HMMs with no control actions update is:

$$m_{1:t+1} = P(e_{t+1}|X_{t+1}) \cdot \max_{x_t} P(X_{t+1}|x_t) \cdot m_{1:t}$$

1. $P(X_{1:t})$
2. $P(U_{1:t})$
3. $P(e_{1:t})$
4. $P(X_{1:t}, U_{1:t} \mid e_{1:t})$
5. $P(X_{1:t}, U_{1:t}, e_{1:t})$
6. $P(U_1)P(X_1 \mid U_1)P(e_1 \mid X_1) \prod_{t=2}^{t} P(U_t \mid U_{t-1})P(X_t \mid U_t)P(e_t \mid X_t)$
7. $P(X_1 \mid U_1)P(e_1 \mid X_1) \prod_{t=2}^{t} P(U_t \mid X_{t-1})P(X_t \mid X_{t-1})P(e_t \mid X_t)$

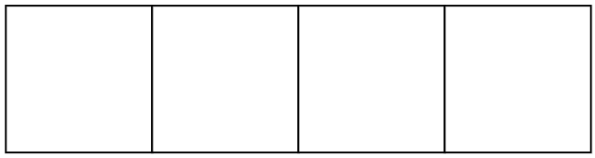
5 1D Gridworld MDP

Consider a 1D gridworld MDP. In this MDP, available actions are *left*, *right*, and *stay*. Stay always results in the agent staying in its current square. Left and right are successful in moving in the intended direction **half of the time**. The other half of the time, the agent stays in its current square. An agent cannot try to move left at the leftmost square, and cannot try to move right on the rightmost square. Staying still on a square gives a reward equivalent to the number on that square and all other transitions give zero reward (meaning any transitions in which the agent moves to a different square give zero reward). Initially, let $V_0(s) = 0, \forall s$ and $\gamma = 0.5$.

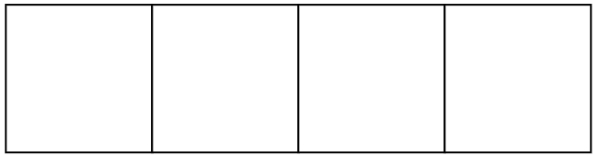


Perform value iteration and put the values you find in the given empty grids.

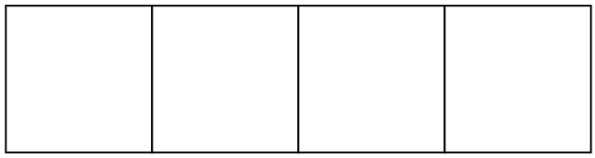
(a) Compute $V_1(s)$



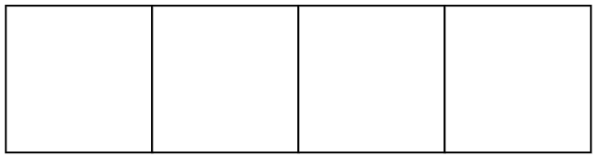
(b) Compute $V_2(s)$



(c) Compute $V^*(s)$.
Hint 1: Think about states where the optimal action is obvious and work from there.
Hint 2: $\sum_{k=0}^{\infty} a^k = 1/(1-a)$, for $0 < a < 1$



(d) What is the optimal policy? Fill each state with the optimal action ("left", "right" or "stay") in the grid below.



6 Discount Shaping

In MDPs, we usually select the discount factor as a hyperparameter then compute the value function. However, although much less common, finding an appropriate discount factor for a desired value could have some use as well (e.g. reverse engineering a real-life process). Consider a gridworld-like MDP with five states $s_1, ..., s_5$. The agent can either stay (a_S) or continue (a_C). You are also given the following for transition kernel $T(s, a, s')$ and reward function $R(s, a)$:

$$T(s_i, a_S, s_i) = 1 \quad \text{for } i \in \{1, 2, 3, 4\}$$

$$T(s_{i+1}, a_C, s_i) = 1 \quad \text{for } i \in \{1, 2, 3, 4\}$$

$$T(s_5, a, s_5) = 1 \quad \text{for all actions } a$$

$$R(s_i, a) = 0 \quad \text{for } i \in \{1, 2, 3, 5\} \text{ and for all actions } a$$

$$R(s_4, a_S) = 0, R(s_4, a_C) = 10$$

If the desired optimal value for state s_1 is $V^*(s_1) = 1$, what is the discount factor γ?

7 Reinforcement Learning I

Imagine an unknown game which has only two states {A, B} and in each state the agent has two actions to choose from: {Up, Down}. Suppose a game agent chooses actions according to some policy π and generates the following sequence of actions and rewards in this unknown game:

t	s_t	a_t	s_{t+1}	r_t
0	A	Down	B	8
1	B	Down	B	-4
2	B	Up	B	0
3	B	Up	A	2
4	A	Up	A	-2

Assume a discount factor $\gamma = 0.5$ and a learning rate $\alpha = 0.5$.

(a) In model-based reinforcement learning, we first estimate the transition function $T(s, a, s')$ and the reward function $R(s, a, s')$. Fill in the following estimates of T and R, obtained from the experience above. Write "n/a" if not applicable or undefined.

$\hat{T}(A, Up, A) =$ ____ $\hat{R}(A, Up, A) =$ ____

$\hat{T}(A, Up, B) =$ ____ $\hat{R}(A, Up, B) =$ ____

$\hat{T}(B, Up, A) =$ ____ $\hat{R}(B, Up, A) =$ ____

$\hat{T}(B, Up, B) =$ ____ $\hat{R}(B, Up, B) =$ ____

(b) Recall the update function of Q-learning is:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a')) \tag{1}$$

Assume that all Q-values initialized as 0. What are the following Q-values learned by running Q-learning with the above experience sequence?

Q(A, Down) = ____ Q(B, Up) = ____

8 Approximate Q-Learning

Suppose that one weekend, you decide to go to an amusement park with your friends. You start at the amusement park feeling well, receiving positive rewards from rides, with some rides offering greater rewards than others. However, each ride carries a risk of making you feel sick. If you continue to go on rides while feeling sick, there is a chance you may recover, but the rewards you earn from rides will likely be reduced and could even be negative.

You have no prior experience visiting amusement parks, so you are unsure about the exact rewards you will receive from each ride (whether feeling well or sick). Similarly, you do not know the probability of getting sick on any particular ride or recovering while sick. What you do know about the rides is summarized below:

Actions / Rides	**Type**	**Wait**	**Speed**
Big Dipper	Rollercoaster	Long	Fast
Wild Mouse	Rollercoaster	Short	Slow
Hair Raiser	Drop tower	Short	Fast
Moon Ranger	Pendulum	Short	Slow
Leave the Park	Leave	Short	Slow

Let's formulate this problem as a two state MDP: well and sick. The actions are the choice of what ride to take. 'Leave the Park' action terminates the run. Taking a ride can either transition to the same state with some probability or take you to the other state. We'll use a feature-based approximation to the Q-values, defined by the following four features and associated weights:

Features	**Initial Weights**
f_0(state, action) = 1 (this is a bias feature that is always 1)	$w_0 = 1$
f_1(state, action) $= \begin{cases} 1 & \text{if action type is Rollercoaster} \\ 0 & \text{otherwise} \end{cases}$	$w_1 = 2$
f_2(state, action) $= \begin{cases} 1 & \text{if action wait is Short} \\ 0 & \text{otherwise} \end{cases}$	$w_2 = 1$
f_3(state, action) $= \begin{cases} 1 & \text{if action speed is Fast} \\ 0 & \text{otherwise} \end{cases}$	$w_3 = 0.5$

(a) Calculate $Q(Well, BigDipper)$.

(b) Apply a Q-learning update based on the sample $(Well, BigDipper, Sick, -10.5)$, using a learning rate of $\alpha = 0.5$ and discount of $\gamma = 0.5$. What are the new weights?

(c) Using our approximation, are the Q-values that involve the sick state the same or different from the corresponding Q-values that involve the well state? In other words, is $Q(Well, a) = Q(Sick, a)$ for any possible action a? Why or why not? (in just one sentence)

(d) Assume we have the original weights from the table on the previous page. What action will an ϵ-greedy approach choose from the $Well$ state? If multiple actions could be chosen, give each action and its probability.

9 Deep RL - Value-based Methods

Answer the questions about Deep Q Networks (DQN), a value-based method, below.

(a) Briefly explain the three key differences between (tabular) Q-learning and DQN.

(b) We usually use deep neural networks to approximate the Q-function in DQNs. In this context, which one of the three differences you explained above do you think improves the performance? Explain your reasoning.

(c) The update frequency of the target network is an important hyperparameter in the training of DQNs. Suppose that while training your agent, you set the update frequency to update the target very rarely, say every 2^{64} steps. How do you think would this impact your training and agent performance?

10 Deep RL - Policy Gradient and Actor-Critic Methods

Answer the questions about policy gradient- and actor-critic-based deep RL methods below.

(a) In the context of policy gradient methods in reinforcement learning, the variance of gradient estimates can significantly affect the learning process. One common technique to reduce variance in policy gradient methods involves leveraging the principle of causality.

Explain how the concept of causality is utilized in policy gradient methods to decrease the variability in the calculated gradients, either through mathematical expressions or verbal explanation.

(b) In actor-critic reinforcement learning methods, the system is comprised of two main components: the actor, which dictates the policy $\pi_\theta(a|s)$, and the critic, which estimates the value function $V_\phi(s)$. Both components are parameterized by θ and ϕ, respectively.

How the value function estimated by the critic implicitly serves as a baseline to improve the gradient estimates for the actor's policy?

(c) Compare the actor-critic and policy gradient methods in terms of bias and variance. Fill in the table below with 'High' or 'Low' to indicate the level of bias or variance you associate with each method and provide a brief (∼1-2 sentence) explanation for each of your answers.

Method	Bias Level	Variance Level
Actor-Critic		
Policy Gradient		