CS 747 (Autumn 2023)
End-semester Examination

Instructor: Shivaram Kalyanakrishnan

5.30 p.m. – 8.30 p.m., November 25, 2023, LA 201 and LA 202

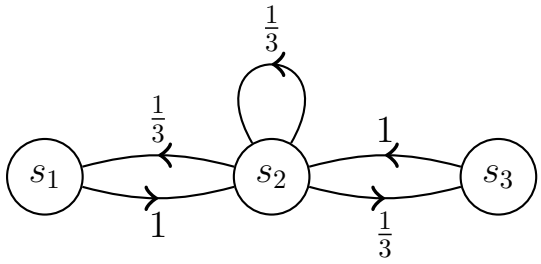
Note. This exam has 8 questions, given on the pages following this one. Provide justifications/calculations/steps along with each answer to illustrate how you arrived at the answer. You will not receive credit for giving an answer without sufficient explanation.

Steps for submission.

1. Bring your phone in a pouch or bag, and keep it on the table you are using.
2. Before the exam begins, turn on "flight mode" on the phone, so it cannot communicate. Do not touch the phone while you are writing the exam.
3. When you are finished writing, put your pen away and stand up.
4. Remain standing while retrieving your phone, scanning your paper, turning off "flight mode", then uploading the scanned pdf to Moodle.
5. You will get 15 extra minutes after the test end time for scanning and uploading (you can do it earlier if you have finished). If you are unable to scan and upload your paper, you will be given a slot later to do so.
6. Before leaving, you must turn in your answer paper to the invigilators in the room.
7. We will only evaluate submissions for which the scanned copy matches the physical answer paper that has been turned in.

Question 1. Consider a continuing MDP with states s_1, s_2, s_3, on which the non-zero transition probabilities on following a fixed policy π are listed in the table below and also shown as annotations in a state transition diagram.

Transition	Probability
$s_1 \to s_2$	1
$s_2 \to s_1$	1/3
$s_2 \to s_2$	1/3
$s_2 \to s_3$	1/3
$s_3 \to s_2$	1



Denote the probability of being in state s_i at time step $t \geq 0$ by p_i^t for $i \in \{1, 2, 3\}$.

1a. Obtain expressions for p_1^t, p_2^t, and p_3^t for $t \geq 1$ in terms of the initial probabilities p_1^0, p_2^0, and p_3^0. You could potentially simplify your calculations by using symmetry found within the problem. [4 marks]

1b. What is $\lim_{t \to \infty} p_i^t$ for $i \in \{1, 2, 3\}$? [1 mark]

Question 2. Consider an episodic MDP $M = (S, A, T, R, \gamma)$, with notations as usual, and with $\gamma = 1$ (that is, no discounting). Suppose that a policy $\pi : S \to A$ is run on M for $N \geq 2$ episodes. From the data (that is, episodic state-action-reward sequences) thereby obtained, let $\hat{V}^N_{\text{First-visit}}$ denote the first-visit Monte Carlo estimate of V^π. Similarly, let $\hat{V}^N_{\text{Every-visit}}$ denote the every-visit Monte Carlo estimate of V^π. Assume that in the data collected, each state $s \in S$ is visited at least once—so the estimators are well-defined.

2a. Write down expressions for $\hat{V}^N_{\text{First-visit}}(s)$ and $\hat{V}^N_{\text{Every-visit}}(s)$ for $s \in S$. You can use the relevant notation from class, or use your own notation along with clear descriptions. [2 marks]

2b. Suppose M is such that every reward is positive: that is, $R(s, a, s') > 0$ for $s, s' \in S, a \in A$. Also, there is no stochasticity in generating rewards. Is it implied that for all $s \in S$, $\hat{V}^N_{\text{First-visit}}(s) \geq \hat{V}^N_{\text{Every-visit}}(s)$? Give a proof to support your claim. [2 marks]

Question 3. Consider an episodic MDP with small, finite sets of states and actions, such as one of the grid world tasks in the textbook by Sutton and Barto (2018). Each episode begins with a non-terminal state selected uniformly at random; termination is guaranteed with probability 1 on each episode regardless of the actions taken at each step. Values are taken to be the expected cumulative reward per episode, with no discounting.

Write down pseudocode for Sarsa(0), implemented such that the agent will converge to optimal behaviour. In other words, if V_i is the expected cumulative reward of the algorithm on episode $i \geq 1$, and $V^\star$ is the optimal expected cumulative reward per episode, then $\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} V_i = V^\star$. You need not provide the proof of optimality. [4 marks]

Question 4. Suppose that a sampling model is available for an episodic task, and the agent performs decision-time planning using this sampling model in order to decide which action to take at each state. Let $\pi_{\text{behaviour}}$ be the policy that results from doing so. In other words, at each state s, the agent computes its action $\pi_{\text{behaviour}}(s)$ by performing decision-time planning.

Concretely, assume that the agent uses the UCT algorithm for decision-time planning. Suppose the agent employs a roll-out policy π_{rollout} internally within UCT. During UCT, the height of the tree kept in memory is $h \geq 1$, and a total of $N \geq 1$ trajectories are generated, starting from s.

What is the relation between π_{rollout} and $\pi_{\text{behaviour}}$ (does one dominate the other)? Consider both theoretical and practical aspects, including the roles of N and h, if any. [4 marks]

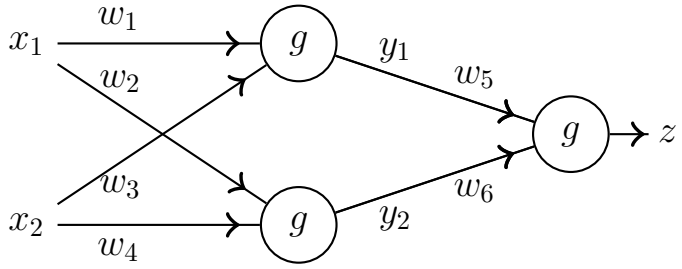
Question 5. A neural network with a total of 6 weights is used to encode a policy for a particular episodic task. There are two features of state, which come in as inputs $x_1, x_2 \in \mathbb{R}$. These inputs are linearly combined and then passed through a sigmoid function to produce the outputs of the two nodes in the hidden layer. In particular, the sigmoid function $g : \mathbb{R} \to [0, 1]$ is given by $g(v) = \frac{1}{1+e^{-v}}$ for $v \in \mathbb{R}$. The outputs from the hidden layer are

$$y_1 = g(w_1 x_1 + w_3 x_2);$$
$$y_2 = g(w_2 x_1 + w_4 x_2).$$

In turn, y_1 and y_2 are linearly combined and passed through g to obtain output $z \in [0, 1]$, as

$$z = g(w_5 y_1 + w_6 y_2).$$

Thus, the parameters of the neural network (shown below) are the weights $w_1, w_2, w_3, w_4, w_5, w_6$.



The neural network implements a policy for a task with two actions: a_1 and a_2. For any given state s, its two features $x_1(s)$ and $x_2(s)$ are provided as input to the neural network. The output z that is computed is taken as the probability of executing action a_1. Hence, the probability of executing action a_2 is $1 - z$.

Suppose that on episode $i \geq 0$, the weights of the neural network are $w_1^i, w_2^i, w_3^i, w_4^i, w_5^i, w_6^i \in \mathbb{R}$. By following the stochastic policy thus encoded, the agent generates a trajectory

$$s^0, a^0, r^0, s^1, a^1, r^1, \ldots, s^T,$$

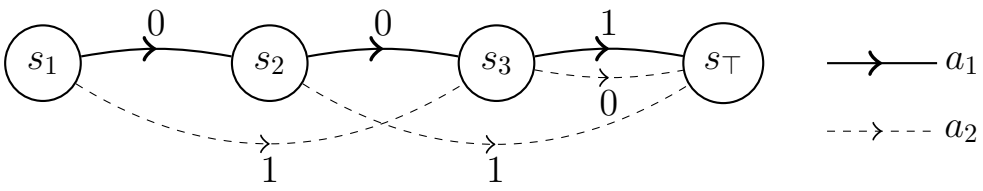
which is a state-action-reward sequence with time step as suffix. s^0 is the starting state, whose value is sought to be maximised, and s^T is the terminal state. If the agent applies REINFORCE based on this trajectory, using learning rate $\alpha \in (0, 1)$, what are the corresponding weights $w_1^{i+1}, w_2^{i+1}, w_3^{i+1}, w_4^{i+1}, w_5^{i+1}, w_6^{i+1}$ after performing the update? You can use $x_1(\cdot)$, $x_2(\cdot)$, and $g(\cdot)$ in your answer. You can also define and use your own functions. [5 marks]

Question 6. The following six episodes of data (each episode given as a state-action-reward sequence) are gathered on an MDP with non-terminal states s_1, s_2, s_3; a terminal state $s_\top$; and actions a_1, a_2. Episodes can begin at any of the non-terminal states. Also, the reward function is deterministic. No discounting is used (that is, $\gamma = 1$).

$s_1, a_1, 3, s_2, a_1, 1, s_3, a_2, 2, s_\top$
$s_1, a_2, 0, s_3, a_1, 4, s_\top$
$s_2, a_1, 0, s_\top$
$s_2, a_2, 4, s_2, a_2, 2, s_3, a_1, 4, s_\top$
$s_2, a_2, 2, s_3, a_2, 2, s_\top$
$s_2, a_2, 2, s_3, a_1, 4, s_\top$

Suppose Experience Replay is run on this data to convergence, using a tabular representation of the state-action space, with appropriate annealing of the learning rate. What is the policy produced by the procedure? Show the steps to arrive at your answer. [4 marks]

Question 7. An episodic MDP with non-terminal states s_1, s_2, s_3; terminal state $s_\top$; and actions a_1 and a_2 is shown below. All transitions are deterministic. Action a_1 is shown using solid arrows; action a_2 is shown using dashed arrows. Rewards for transitions are annotated on the corresponding arrows.



The starting state for each episode is selected uniformly at random among s_1 and s_2. Unfortunately, the agent interacting with this MDP is unable to sense state. However, the agent can remember the number of time steps that have elapsed thus far in the episode; it can also use randomness in action-selection.

What is the maximum (undiscounted) expected cumulative reward that the agent can accrue in each episode? What "policy" must it follow in order to do so? [5 marks]

Question 8. In the class lectures, we covered two applications: "Autonomous helicopter flight via reinforcement learning" (Ng et al., 2003) and "Mastering the game of Go with deep neural networks and tree search" (Silver et al., 2016). The latter paper introduces AlphaGo (*not* AlphaGo Zero or AlphaZero). For *each* of these applications, specify which among the following methods/aspects are a part of the solution, with brief supporting explanation.

A. Supervised learning
B. Data generated by humans
C. Direct optimisation of policy (by policy gradient or policy search)
D. Decision-time planning
E. Value function learning
F. Neural networks

[4 marks]

Solutions

1a. For $t \geq 1$, we observe from the MDP that

$$\begin{aligned} p_1^t &= \frac{1}{3} p_2^{t-1}, \\ p_2^t &= p_1^{t-1} + \frac{1}{3} p_2^{t-1} + p_3^{t-1}, \\ p_3^t &= \frac{1}{3} p_2^{t-1}. \end{aligned}$$

Since $p_{t-1}^1 + p_{t-1}^2 + p_{t-1}^3 = 1$, we obtain

$$p_2^t = 1 - \frac{2}{3} p_2^{t-1},$$

which on expanding yields

$$p_2^t = 1 + \frac{-2}{3} + \left(\frac{-2}{3}\right)^2 + \cdots + \left(\frac{-2}{3}\right)^{t-1} + \left(\frac{-2}{3}\right)^t p_2^0.$$

This expression (which contains an arithmetic proression) simplifies to

$$p_2^t = \frac{3}{5}\left(1 - \left(\frac{-2}{3}\right)^t\right) + \left(\frac{-2}{3}\right)^t p_2^0.$$

Also observe that $p_1^t = p_3^t$, in turn equal to $\frac{1}{2}(1 - p_2^t)$. Thus

$$p_1^t = p_3^t = \frac{1}{5} + \frac{3}{10}\left(\frac{-2}{3}\right)^t - \frac{1}{2}\left(\frac{-2}{3}\right)^t p_2^0.$$

1b. Since $\left|\frac{2}{3}\right| < 1$, $\lim_{t\to\infty} \left(\frac{-2}{3}\right)^t = 0$. Consequently, we get

$$\begin{aligned} \lim_{t\to\infty} p_1^t &= \frac{1}{5}, \\ \lim_{t\to\infty} p_2^t &= \frac{3}{5}, \\ \lim_{t\to\infty} p_3^t &= \frac{1}{5}. \end{aligned}$$

2a. We have the following definitions from the class lectures. For $s \in S$, $i \geq 1, j \geq 1$, let $\mathbf{1}(s,i,j)$ be 1 if s is visited at least j times on episode i (else $\mathbf{1}(s,i,j) = 0$), and $G(s,i,j)$ be the discounted long-term reward starting from the j-th visit of s on episode i.

$$\hat{V}^N_{\text{First-visit}}(s) = \frac{\sum_{i=1}^N G(s,i,1)}{\sum_{i=1}^N \mathbf{1}(s,i,1)}.$$

$$\hat{V}^N_{\text{Every-visit}}(s) = \frac{\sum_{i=1}^N \sum_{j=1}^\infty G(s,i,j)}{\sum_{i=1}^N \sum_{j=1}^\infty \mathbf{1}(s,i,j)}.$$

Notice that we are given that both denominators are positive.

2b. It is not true that the every-visit estimate is necessarily upper-bounded by the first-visit estimate. We show a counterexample, in which $N = 2$ episodes of data are gathered. As state-action-reward sequences, they are as follows.

Episode 1: $s_1, a_1, 10, s_1, a_1, 10, s_2, a_1, 20, s_\top$.

Episode 2: $s_1, a_1, 1, s_3, a_1, 1, s_\top$.

The calculations below show that $\hat{V}^N_{\text{First-visit}}(s_1) < \hat{V}^N_{\text{Every-visit}}(s_1)$.

$$\hat{V}^N_{\text{First-visit}}(s_1) = \frac{(10+10+20)+(1+1)}{2} = 21;$$
$$\hat{V}^N_{\text{Every-visit}}(s_1) = \frac{(10+10+20)+(10+20)+(1+1)}{3} = 24.$$

3. Pseudocode for Sarsa(0) is provided below. Qualitatively, the main aspects to ensure are that (1) the policy followed is GLIE; (2) the learning rate is annealed properly; (3) the (s, a, r, s', a') update is done *after* a' is selected to be taken; (4) the target is r when s' is a terminal state; and (5) no update happens across episodes—such as with s from one episode and s' from the next episode.

$Q \leftarrow \mathbf{0}$.
$t \leftarrow 0$. //Overall time steps.
$e \leftarrow 0$. //Overall episodes.
For each episode:
    The agent is born in state s.
    $a \leftarrow$ NULL.
    Do:
        If a is NULL:
            Sample $a \sim \pi_{Q,t,e}(s)$.
        Take action a, obtain reward r, next state s'.
        If s' is terminal:
            $Q(s, a) \leftarrow Q(s, a) + \alpha_{t,e}\{r - Q(s, a)\}$.
        Else:
            Sample $a' \sim \pi_{Q,t,e}(s')$.
            $Q(s, a) \leftarrow Q(s, a) + \alpha_{t,e}\{r + \gamma Q(s', a') - Q(s, a)\}$.
            $a \leftarrow a'$.
        $s \leftarrow s'$.
        $t \leftarrow t + 1$.
    While s is not terminal.
    $e \leftarrow e + 1$.

The policy $\pi_{Q,t,e}$ can be taken to be ϵ-greedy with respect to Q, where we have the flexibility of annealing ϵ either after each episode or after each time step. So, for example, we could take $\epsilon = \frac{1}{1+e}$ or $\frac{1}{1+t}$. Similarly, the learning rate $\alpha_{t,e}$ can also be taken as either $\frac{1}{1+e}$ or $\frac{1}{1+t}$. Other choices are also fine as long as exploration is GLIE, and the learning rate satisfies Robbins and Monro's conditions.

4. The conceptual basis of UCT is that if N is infinite, then

$$\pi_{\text{behaviour}}(s) = \operatorname*{argmax}_{a \in A} \sum_{s' \in S} T(s,a,s')\{R(s,a,s') + \gamma (B^\star)^{h-1}(V^{\pi_{\text{rollout}}})(s')\}.$$

If $h = 1$, notice that we obtain $\pi_{\text{behaviour}}(s) = \text{argmax}_{a \in A}\, Q^{\pi_{\text{rollout}}}(s,a)$, which implies (by the policy improvement theorem) that $\pi_{\text{behaviour}} \succ \pi_{\text{rollout}}$ if π_{rollout} is not optimal, and otherwise both policies are optimal. The same result holds for larger values of h.

In practice, we are constrained to run UCT with finite values of N and h. Hence, although the claim above is not necessarily true, it will still hold with a probability determined by N, h and the parameters of the MDP. We also expect that typically in practice, larger values of h will give rise to behaviour policies with higher returns. Notice that if h and N are both infinite, then $\pi_{\text{behaviour}}$ must be an optimal policy regardless of π_{rollout}.

5. Let $\bar{w}$, a vector, be a short form for $(w_1, w_2, w_3, w_4, w_5, w_6)$. We define the following functions to help with our update from $\bar{w}^i$ to $\bar{w}^{i+1}$.

$$
\begin{aligned}
y_1(\bar{w}, s) &= g(w_1 x_1(s) + w_3 x_2(s)).\\
y_2(\bar{w}, s) &= g(w_2 x_1(s) + w_4 x_2(s)).\\
z(\bar{w}, s) &= g(w_5 y_1(\bar{w}, s) + w_6 y_2(\bar{w}, s)).\\
g'(v) &= \frac{e^{-v}}{(1+e^{-v})^2}.\\
\frac{\partial z(\bar{w}, s)}{\partial w_5} &= g'(w_5 y_1(\bar{w}, s) + w_6 y_2(\bar{w}, s)) y_1(\bar{w}, s).\\
\frac{\partial z(\bar{w}, s)}{\partial w_6} &= g'(w_5 y_1(\bar{w}, s) + w_6 y_2(\bar{w}, s)) y_2(\bar{w}, s).\\
\frac{\partial z(\bar{w}, s)}{\partial w_1} &= g'(w_5 y_1(\bar{w}, s) + w_6 y_2(\bar{w}, s)) w_5 g'(w_1 x_1(s) + w_3 x_2(s)) x_1(s).\\
\frac{\partial z(\bar{w}, s)}{\partial w_2} &= g'(w_5 y_1(\bar{w}, s) + w_6 y_2(\bar{w}, s)) w_6 g'(w_2 x_1(s) + w_4 x_2(s)) x_1(s).\\
\frac{\partial z(\bar{w}, s)}{\partial w_3} &= g'(w_5 y_1(\bar{w}, s) + w_6 y_2(\bar{w}, s)) w_5 g'(w_1 x_1(s) + w_3 x_2(s)) x_2(s).\\
\frac{\partial z(\bar{w}, s)}{\partial w_4} &= g'(w_5 y_1(\bar{w}, s) + w_6 y_2(\bar{w}, s)) w_6 g'(w_2 x_1(s) + w_4 x_2(s)) x_2(s).
\end{aligned}
$$

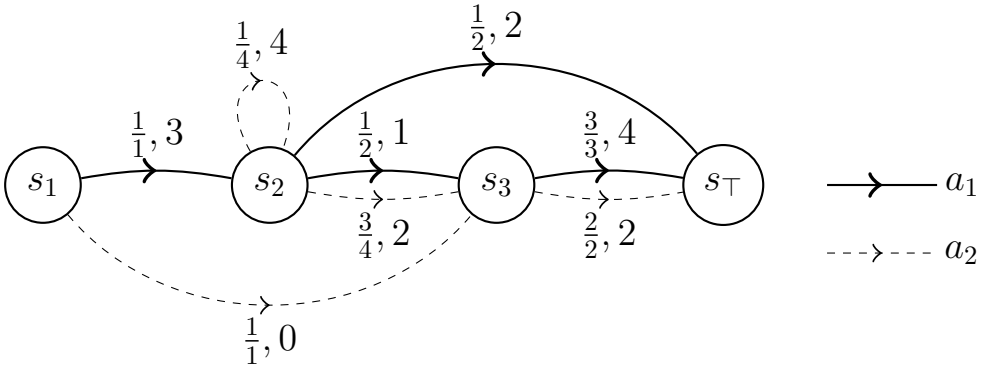
With these definitions done, we obtain the following for $k \in \{1, 2, 3, 4, 5, 6\}$.

$$
\begin{aligned}
\pi(\bar{w}, s, a) &= \begin{cases} z(\bar{w}, s) & \text{if } a = a_1, \\ 1 - z(\bar{w}, s) & \text{if } a = a_2. \end{cases}\\
\frac{\partial \pi(\bar{w}, s, a)}{\partial w_k} &= \begin{cases} \frac{\partial z(\bar{w}, s)}{\partial w_k} & \text{if } a = a_1, \\ -\frac{\partial z(\bar{w}, s)}{\partial w_k} & \text{if } a = a_2. \end{cases}
\end{aligned}
$$

Finally we have the REINFORCE update rule. For $k \in \{1, 2, 3, 4, 5, 6\}$,

$$
\bar{w}_k^{i+1} = w_k^i + \alpha \sum_{t=0}^{T-1} \left(\frac{\partial \pi(\bar{w}^i, s^t, a^t)}{\partial w_k} \right) \left(\frac{1}{\pi(\bar{w}^i, s^t, a^t)} \right) \left(\sum_{j=t}^{T-1} r^j \right).
$$

6. The maximum likelihood MDP $\widehat{M}$ to generate the observed data is obtained by calculating the empirical frequency (and reward) of each transition. It works out to the MDP shown below. Solid arrows correspond to action a_1, and dashed arrows correspond to action a_2. Annotations are of the form "transition probability, reward"; zero-probability transitions are not included. Notice, for example, that that (s_2, a_2) occurs four times in the data, of which one transition is into s_2, and the remaining three into s_3. Hence, the empirical probabilities are $\frac{1}{4}$ for (s_2, a_2, s_2), and $\frac{3}{4}$ for (s_2, a_2, s_3).



At convergence, Experience Replay (which updates Q values) will converge to $\hat{Q}^\star_{\widehat{M}}$, which induces policy $\pi^\star_{\widehat{M}}$. The steps below are based on the transition probabilities and rewards in the empirical MDP $\widehat{M}$. The step to calculate $Q^\star_{\widehat{M}}(s_2, a_2)$ depends on $V^\star_{\widehat{M}}(s_2)$, which has to be the maximum of $Q^\star_{\widehat{M}}(s_2, a_1)$ and $Q^\star_{\widehat{M}}(s_2, a_2)$. Upon trying with both values, it is seen that the latter is larger.

$$
\begin{aligned}
Q^\star_{\widehat{M}}(s_3, a_1) &= 4. \\
Q^\star_{\widehat{M}}(s_3, a_2) &= 2. \\
V^\star_{\widehat{M}}(s_3) &= 4. \\
\pi^\star_{\widehat{M}}(s_3) &= a_1. \\
Q^\star_{\widehat{M}}(s_2, a_1) &= \frac{1}{2}(2+0) + \frac{1}{2}(1 + V^\star_{\widehat{M}}(s_3)) = \frac{7}{2}. \\
Q^\star_{\widehat{M}}(s_2, a_2) &= \frac{1}{4}(4 + V^\star_{\widehat{M}}(s_2)) + \frac{3}{4}(2 + V^\star_{\widehat{M}}(s_3)) = \frac{22}{3}. \\
V^\star_{\widehat{M}}(s_2)) &= \frac{22}{3}. \\
\pi^\star_{\widehat{M}}(s_2) &= a_2. \\
Q^\star_{\widehat{M}}(s_1, a_1) &= 3 + V^\star_{\widehat{M}}(s_2) = \frac{31}{3}. \\
Q^\star_{\widehat{M}}(s_1, a_2) &= 0 + V^\star_{\widehat{M}}(s_3) = \frac{7}{2}. \\
V^\star_{\widehat{M}}(s_1) &= \frac{31}{3}. \\
\pi^\star_{\widehat{M}}(s_1) &= a_1.
\end{aligned}
$$

7. As specified in the question, if the agent can only remember time step, the most expressive class of policies it can optimise over is mappings from the set of possible time steps elapsed to the set of probability distributions over actions. In other words, the agent must optimise probabilities $x, y, z \in [0, 1]$ where x is the probability of taking a_1 at $t = 0$, y is the probability of taking a_1 at $t = 1$, and z is the probability of taking a_1 at $t = 2$. In short, the agent has to optimise over this template.

Memory	Probability of taking a_1
$t = 0$	x
$t = 1$	y
$t = 2$	z

We evaluate this policy on the MDP, starting from each state (which the agent itself might not unambiguously know while acting) and for each "memory state" (in this case a time step), denoted as superscript.

$$
\begin{aligned}
V^2(s_3) &= z(1) + (1 - z)(0) = z. \\
V^1(s_3) &= y(1) + (1 - y)(0) = y. \\
V^0(s_3) &= x(1) + (1 - x)(0) = x. \\
V^1(s_2) &= y(0 + V^2(s_3)) + (1 - y)(1 + 0) = yz + 1 - y. \\
V^0(s_2) &= x(0 + V^1(s_3)) + (1 - x)(1 + 0) = xy + 1 - x. \\
V^0(s_1) &= x(0 + V^1(s_2)) + (1 - x)(1 + V^1(s_3)) = xyz - 2xy + 1 + y.
\end{aligned}
$$

If the agent is initialised uniformly at random at s_1 or s_2, the scalar value of policy (x, y, z) is

$$\frac{1}{2}(V^0(s_1) + V^0(s_2)) = xyz - xy + 2 - x + y.$$

Since x, y, z are all non-negative, first we notice that $z = 1$ will always optimise the value. After setting $z = 1$, we notice that the value decreases with x and increases with y. Hence an optimal setting is $x = 0, y = 1, z = 1$, which yields a value of $\frac{3}{2}$. In fact, if $x = 0$ and $y = 1$, the agent never lives for more than 2 time steps—that is, never has to take an action at $t = 2$—which means z can be set arbitrarily to obtain the same value of $\frac{3}{2}$.

An alternative assumption that can be made (not explicitly specified in the question) is that the agent can also remember the sequence of *actions* it has taken thus far in the episode. In this case, the agent has to optimise probabilities $p, p_1, p_2, p_{11}, p_{12}, p_{21}, p_{22}$, as listed below.

Memory	Probability of taking a_1
$\emptyset$	p
a_1	p_1
a_2	p_2
a_1, a_1	p_{11}
a_1, a_2	p_{12}
a_2, a_1	p_{21}
a_2, a_2	p_{22}

In general this more expressive class of policies can provide better returns. However, on this particular example, the maximum value achievable remains $\frac{3}{2}$ (we leave it as en exercise to the student to work out this out).

8. The work on helicopter control uses [A] supervised learning of the model, based on [B] data logged by a human controller. [C] The PEGASUS method for policy search is performed using hill climbing. [F] The policy is represented as a neural network. The two items which are *not* relevant to the helicopter control work are [D] and [E].

AlphaGo uses [A] supervised learning on [B] games of Go played by human experts. The key learning algorithm is [C] REINFORCE, which is a policy gradient algorithm. [D] UCT, which is a decision-time planning algorithm, is deployed while playing th game. [E] A value function is learned to evaluate leaves during decision-time planning. [F] Both the policy and the value function are represented using neural networks.