CS 747 (Autumn 2021): Weekly Quizzes

Instructor: Shivaram Kalyanakrishnan

October 21, 2021

Note. 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.

Submission. Write down your answer by hand, then scan and upload to Moodle. Write clearly and legibly. Be sure to mention your roll number.

Week 11

Question. An agent interacting with episodic MDP (S, A, T, R, γ) executes a stochastic policy π parameterised by a d-dimensional parameter vector $w \in \mathbb{R}^d$ for some $d \geq 1$. States are represented by features $\phi_i : S \to \mathbb{R}$, $1 \leq i \leq d$. The MDP has two actions: $A = \{0, 1\}$. With parameter vector $w = (w_1, w_2, \ldots, w_d)$, the policy π is specified by:

$$\pi_w(s, 0) = \frac{1}{1 + e^{-\sum_{i=1}^{d} w_i \phi_i(s)}}; \qquad \pi_w(s, 1) = 1 - \pi(s, 0).$$

Following policy π_w, suppose the agent encounters the trajectory

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

where s^T is a terminal state. Also suppose that the agent updates its policy parameters from w to w' by performing a REINFORCE update based on this episode, with learning rate $\alpha > 0$. There is no baseline subtraction.

Write down pseudocode to compute each element $w'_i, 1 \leq i \leq d$, based on w and the recorded trajectory. Do not perform operations directly on vectors; instead perform updates to scalars, iterating over them if necessary. You can assume that $\pi_w(s, a)$ and $\phi_i(s)$ are already available in memory for $s \in S, a \in A, i \in \{1, 2, \ldots, d\}$: you can use them as variables in your code. [5 marks]

Week 10

Question. This week's questions are related to function approximation and policy search.

a. Imagine a task with a small number of states (say a few tens or hundreds) and actions (say single digit), such as one of the many "grid world" examples given in the textbook by Sutton and Barto (2018). If an agent has to learn successful behaviour on such tasks, it is quite conceivable to use the usual "tabular" representation, with one entry for each Q-value, and to converge to optimal behaviour. Even so, can you see any advantages of using generalisation, under which different states or state-action pairs would *share* parameters that get updated during learning? [1 mark]

b. Grid search is a conceptually simple form of policy search. Suppose the policy for a particular task is parameterised by d parameters $w_1, w_2, \ldots, w_d \in [0, 1]$ for some $d \geq 1$. In grid search, the parameter ranges are discretised, say into $m \geq 2$ values each, and only parameters vectors that take these values are evaluated. Hence, the policies evaluated are the ones parameterised by $w_1, w_2, \ldots, w_d \in \{0, \frac{1}{m-1}, \frac{2}{m-1}, \ldots, \frac{m-2}{m-1}, 1\}$. Each such policy is evaluated by performing $L \geq 1$ rollouts from the designated start state (since, in general, the task could be stochastic). The empirical average of the (discounted) long-term value in the rollouts is taken as the value of the corresponding policy. The output of the search is the discrete-valued parameter vector that registered the highest empirical value. Comment on the role of the hyperparameters d, m, and L on the efficacy as well as the practicality of performing grid search. [2 marks]

Solution.
a. Generalisation can prompt faster learning. On many naturally-occurring tasks, the features we encode impart the bias that "similar states have similar values"—which can help generate good estimates even for unvisited states. Especially if we are bootstrapping, this bias can accelerate the learning process.

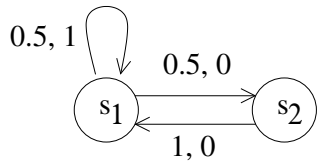
b. Although it depends on how the parameters are put to use, for most function approximators we can expect that having more parameters (a larger setting of d) can allow for the representation of more complex policies. A larger value of m means we are conducting a search over a larger number of policies—potentially increasing the maximum value we will obtain from the search. Unfortunately the number of discrete-valued policies is m^d, which would be feasible to exhaustively evaluate only for small values of d. For tasks with a high amount of stochasticity, L would need to be large to blunt out the noise and identify the true maximising policy. With small values of L, we can end up with a poor policy even if we have searched over a large number of policies. The number of samples would scale linearly with L.

Week 9

Question. This question relates to a prediction problem using linear function approximation. A 2-state MDP and a fixed policy π result in the following transitions:

- $s_1 \to s_1$ with probability $\frac{1}{2}$ and reward 1;
- $s_1 \to s_2$ with probability $\frac{1}{2}$ and reward 0;
- $s_2 \to s_1$ with probability 1 and reward 0.

The task, depicted below, uses a discount factor $\gamma = \frac{1}{2}$.



We aim to approximate V^π using a single scalar parameter $w \in \mathbb{R}$, such that V^π is approximated well by $\hat{V}_w$. In particular, s_1 has an associated feature value $\phi(s_1) = 3$, and s_2 has an associated feature value $\phi(s_2) = 1$. We set $\hat{V}_w(s) = w\phi(s)$ for $s \in \{s_1, s_2\}$, and aim to find w such that $\hat{V}_w \approx V^\pi$.

a. Solve for

$$w^\star = \operatorname*{argmin}_{w \in \mathbb{R}} \sum_{i=1}^{2} \mu^\pi(s_i)(V^\pi(s_i) - \hat{V}_w(s_i))^2,$$

where μ^π denotes the stationary distribution of π. [3 marks]

b. Suppose an agent uses Linear TD(0) to estimate the parameter w. It uses harmonic discounting, hence should be expected to converge. Suppose it converges to w_0. Based the results discussed in class, what is the farthest that w_0 can be from $w^\star$? In other words, give an upper bound on $|w_0 - w^\star|$. [3 marks]

Solution.

a. The Bellman equations for π are

$$\begin{aligned} V^\pi(s_1) &= \frac{1}{2}(1 + \gamma V^\pi(s_1)) + \frac{1}{2}\gamma V^\pi(s_2), \\ V^\pi(s_2) &= \gamma V^\pi(s_1), \end{aligned}$$

which give: $V^\pi(s_1) = \frac{4}{5}, V^\pi(s_2) = \frac{2}{5}$. The stationary distribution μ^π satisfies

$$\begin{aligned} \mu^\pi(s_1) &= \frac{1}{2}\mu^\pi(s_1) + \mu^\pi(s_2), \\ \mu^\pi(s_1) + \mu^\pi(s_2) &= 1, \end{aligned}$$

giving $\mu^\pi(s_1) = \frac{2}{3}, \mu^\pi(s_2) = \frac{1}{3}$. Substituting, we get

$$\begin{aligned} w^\star &= \operatorname*{argmin}_{w\in\mathbb{R}} \left(\frac{2}{3}\left(\frac{4}{5} - 3w\right)^2 + \frac{1}{3}\left(\frac{2}{5} - w\right)^2 \right) \\ &= \operatorname*{argmin}_{w\in\mathbb{R}} \left(2(4-15w)^2 + (2-5w)^2 \right) \\ &= \operatorname*{argmin}_{w\in\mathbb{R}} (475w^2 - 260w) \\ &= \frac{26}{95}. \end{aligned}$$

b. We use the established upper bound:

$$MSVE(w_0) \leq \frac{1}{1-\gamma} MSVE(w^\star)$$

Now,

$$\begin{aligned} MSVE(w_0) &= \sum_{i=1}^{2} \mu^\pi(s_i)(V^\pi(s_i) - \hat{V}_{w_0}(s_i))^2 \\ &= \frac{2}{3}\left(\frac{4}{5} - 3w_0\right)^2 + \frac{1}{3}\left(\frac{2}{5} - w_0\right)^2 \\ &= \frac{1}{75}(475(w_0)^2 - 260w_0 + 36). \end{aligned}$$

Similarly,

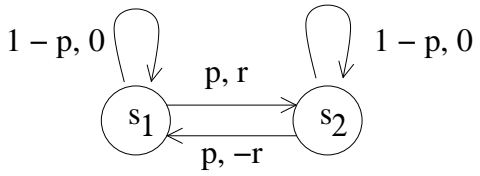
$$MSVE(w^\star) = \frac{1}{75}(475(w^\star)^2 - 260w^\star + 36) = \frac{26\times 26}{15\times 95} - \frac{52\times 26}{15\times 95} + \frac{12}{25} = \frac{12}{25} - \frac{26\times 26}{15\times 95}.$$

Hence, we get

$$\begin{aligned} \frac{1}{75}(475(w_0)^2 - 260w_0 + 36) &\leq 2\left(\frac{12}{25} - \frac{26\times 26}{15\times 95}\right) \\ &\iff \\ (w_0)^2 - \frac{52}{95}w_0 + \frac{36}{475} &\leq \frac{150}{475}\left(\frac{12}{25} - \frac{26\times 26}{15\times 95}\right) \\ &\iff \\ \left(w_0 - \frac{26}{95}\right)^2 &\leq \frac{150}{475}\left(\frac{12}{25} - \frac{26\times 26}{15\times 95}\right) - \frac{36}{475} + \frac{26\times 26}{95\times 95} = \frac{8}{95\times 95} \\ &\iff \\ |w_0 - w^\star| &\leq \frac{2\sqrt{2}}{95}. \end{aligned}$$

Week 8

Question. Consider a prediction problem on an MDP with states s_1 and s_2. The policy π being executed is such that each state transitions to the other with probability $p \in (0, 1)$, and has a self loop with probability $1 - p$. Going from s_1 to s_2 earns a reward of $r > 0$, while going from s_2 to s_1 earns $-r$. The reward for staying in the same state in consecutive time steps is zero. The discount factor γ is set to $\frac{3}{4}$. The figure below provides a pictorial representation.



The TD(0) algorithm is applied to estimate the value function V^π. The initial estimate V^0 is such that $V^0(s_1) = V^0(s_2) = 0$. Suppose the agent starts in state s_1 and performs two actions (hence two learning updates). The first is a TD(0) update with learning rate 1, and the second is a TD(0) update with learning rate $\frac{1}{2}$. Let V^1 denote the value function estimate after the first update, and V^2 the value function estimate after the second update.

a. What is the distribution of V^2? In other words, enumerate all possible values for the pair $(V^2(s_1), V^2(s_2))$, along with the non-zero probability that the pair is realised at the end of the two TD updates. [3 marks]

b. From your answer to part a, work out the values of $\mathbb{E}[V^2(s_1)]$ and $\mathbb{E}[V^2(s_2)]$. [1 mark]

c. Suppose the agent continues with more updates, using a learning rate that is annealed harmonically: $\frac{1}{3}$ for the third update, $\frac{1}{4}$ for the fourth update, and so on. If V^t denotes the value function estimate after t updates for $t \geq 1$, what are the values of $\lim_{t\to\infty} \mathbb{E}[V^t(s_1)]$ and $\lim_{t\to\infty} \mathbb{E}[V^t(s_2)]$? [2 marks]

It should be relatively simple to code up the learning process described above and verify the correctness of your answers by simulation, for different values of p and r. Although not required, you are encouraged to do so to build familiarity with the random process by which learning proceeds.

Solution.

a. There are four possible trajectories that can materialise in 2 steps; we derive the values of V^2 in each case. We use α_t to denote the learning rate for the t-th update, $t \geq 1$. Note that only one state has its value updated at each step; the estimate for the other state remains unaffected.

- With probability $(1 - p)^2$, the transitions are $s_1 \to s_1 \to s_1$. In this case, we get

$$V^1(s_1) = V^0(s_1)(1 - \alpha_1) + \alpha_1(0 + \gamma V^0(s_1)) = 0.$$
$$V^1(s_2) = V^0(s_2) = 0.$$
$$V^2(s_1) = V^1(s_1)(1 - \alpha_2) + \alpha_2(0 + \gamma V^1(s_1)) = 0.$$
$$V^2(s_2) = V^1(s_2) = 0.$$

- With probability $(1-p)p$, the transitions are $s_1 \to s_1 \to s_2$. In this case, we get

$$\begin{aligned}
V^1(s_1) &= V^0(s_1)(1-\alpha_1) + \alpha_1(0 + \gamma V^0(s_1)) = 0. \\
V^1(s_2) &= V^0(s_2) = 0. \\
V^2(s_1) &= V^1(s_1)(1-\alpha_2) + \alpha_2(r + \gamma V^1(s_2)) = \frac{r}{2}. \\
V^2(s_2) &= V^1(s_2) = 0.
\end{aligned}$$

- With probability p^2, the transitions are $s_1 \to s_2 \to s_1$. In this case, we get

$$\begin{aligned}
V^1(s_1) &= V^0(s_1)(1-\alpha_1) + \alpha_1(r + \gamma V^0(s_2)) = r. \\
V^1(s_2) &= V^0(s_2) = 0. \\
V^2(s_1) &= V^1(s_1) = r. \\
V^2(s_2) &= V^1(s_2)(1-\alpha_2) + \alpha_2(-r + \gamma V^1(s_1)) = -\frac{r}{8}.
\end{aligned}$$

- With probability $p(1-p)$, the transitions are $s_1 \to s_2 \to s_2$. In this case, we get

$$\begin{aligned}
V^1(s_1) &= V^0(s_1)(1-\alpha_1) + \alpha_1(r + \gamma V^0(s_2)) = r. \\
V^1(s_2) &= V^0(s_2) = 0. \\
V^2(s_1) &= V^1(s_1) = r. \\
V^2(s_2) &= V^1(s_2)(1-\alpha_2) + \alpha_2(0 + \gamma V^1(s_2)) = 0.
\end{aligned}$$

b. The expected values are obtained by aggregating the four possible cases from part a.

$$\mathbb{E}[V^s(s_1)] = (1-p)^2(0) + (1-p)p\left(\frac{r}{2}\right) + p^2(r) + p(1-p)(r) = \frac{rp(3-p)}{2}.$$

$$\mathbb{E}[V^s(s_2)] = (1-p)^2(0) + (1-p)p(0) + p^2\left(-\frac{r}{8}\right) + p(1-p)(0) = -\frac{p^2 r}{8}.$$

c. The algorithm is guaranteed to converge to V^π, which we can obtain by solving these Bellman equations.

$$\begin{aligned}
V^\pi(s_1) &= p(r + \gamma V^\pi(s_2)) + (1-p)\gamma V^\pi(s_1), \\
V^\pi(s_2) &= p(-r + \gamma V^\pi(s_1)) + (1-p)\gamma V^\pi(s_2).
\end{aligned}$$

Substituting for γ, we obtain

$$\begin{aligned}
\lim_{t\to\infty} \mathbb{E}[V^t(s_1)] &= V^\pi(s_1) = \frac{4pr}{6p+1}, \\
\lim_{t\to\infty} \mathbb{E}[V^t(s_2)] &= V^\pi(s_2) = -\frac{4pr}{6p+1}.
\end{aligned}$$

Week 7

Question. An agent interacts with an episodic MDP (S, A, T, R, γ), where $S = \{1, 2, 3, \top\}$, with $\top$ being the sole terminal state. The agent follows a fixed, deterministic policy $\pi : S \setminus \{\top\} \to A$. For $s \in S \setminus \{\top\}$, $s' \in S$, the probability of transitioning from s to s' under π is $T(s, \pi(s), s')$; for convenience let us denote this quantity $p_{ss'}$. This 12-dimensional "p" vector, induced by π on T,is the main quantity of interest in this question.

Suppose episodes are always started at state 1. Consider the sequence of states h, given by

$$h = (1, 1, 2, 1, 2, 1, 1, 3, 2, 1, 1, 2, 3, \top).$$

a. What is the probability that the sequence of states encountered by the agent in its first episode is h? Provide your answer as a function of the p vector; in other words, $p_{ss'}, s \in S \setminus \{\top\}, s' \in S$ (although not all twelve probabilities need to occur in your answer). [1 mark]

b. The data the agent has seen through its interaction enables it to get some idea about the underlying MDP. In part a, you provided the probability of any arbitrary p vector generating h in the first episode. Let $p^\star$ be a p vector that has the maximum probability (among all p vectors) of generating h. If there is a unique value of $p^\star$, fill its entries in the table below; otherwise fill entries corresponding any maximising p vector. Explain how you arrived at your answer [3 marks].

$p^\star_{11}$	$p^\star_{12}$	$p^\star_{13}$	$p^\star_{1\top}$	$p^\star_{21}$	$p^\star_{22}$	$p^\star_{23}$	$p^\star_{2\top}$	$p^\star_{31}$	$p^\star_{32}$	$p^\star_{33}$	$p^\star_{3\top}$

Solution.
a. We have to multiply the probabilities of going from 1 to 1, then 1 to 2, then 2 to 1, ..., 3 to $\top$, which for a given p vector is

$$(p_{11})^3 (p_{12})^3 (p_{13})(p_{21})^3 p_{23} p_{32} p_{3\top}.$$

b. For $s \in \{1, 2, 3\}$, we know that $p_{s1} + p_{s2} + p_{s3} + p_{s\top} = 1$. Simple calculus establishes that for non-negative integers a, b, c, d, the quantity $(p_{s1})^a (p_{s2})^b (p_{s3})^c (p_{s\top})^d$ has a unique maximum $p^\star$ where

$$p^\star_{s1} = \frac{a}{a+b+c+d}, p^\star_{s2} = \frac{b}{a+b+c+d}, p^\star_{s3} = \frac{c}{a+b+c+d}, p^\star_{s4} = \frac{d}{a+b+c+d}.$$

The vector $p^\star$ that achieves the highest probability of generating h is specified in the table below.

$p^\star_{11}$	$p^\star_{12}$	$p^\star_{13}$	$p^\star_{1\top}$	$p^\star_{21}$	$p^\star_{22}$	$p^\star_{23}$	$p^\star_{2\top}$	$p^\star_{31}$	$p^\star_{32}$	$p^\star_{33}$	$p^\star_{3\top}$
$\frac{3}{7}$	$\frac{3}{7}$	$\frac{1}{7}$	0	$\frac{3}{4}$	0	$\frac{1}{4}$	0	0	$\frac{1}{2}$	0	$\frac{1}{2}$

Week 5

Question. Consider an MDP (S, A, T, R, γ) in which the set of states is $S = \{1, 2, \ldots, n\}$. This question relates to a *variation* of value iteration. Concretely, let us denote the variant given in class CV (for "class variant"), and the one given by the pseudocode below QV (for "quiz variant"). In QV, each state is initialised with value 0. Subsequently the value function V is updated through T iterations; in each iteration all n state values get updated.

QV
$V \leftarrow n$-dimensional $\mathbf{0}$ vector.
For $i = 0, 1, \ldots, T-1$:
  For $s = 1, 2, \ldots, n$:
    $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s')\{R(s, a, s') + \gamma V(s')\}$.
Return V.

a. Pay close attention to the pseudocode. In qualitative terms, what is the main difference between QV and CV? In quantitative terms, derive an operator $B : (S \to \mathbb{R}) \to (S \to \mathbb{R})$ that is being implemented by QV. In other words, for what B can we describe the code above as T successive applications of B to the initial value vector? Feel free to use the Bellman optimality operator $B^\star$, as well as recursion, in your definition of B. [2 marks]

b. QV is used quite commonly in practice, and is known to converge to $V^\star$ (as $T \to \infty$). Your job is to prove this to be the case. To that end, show that (1) B is a contraction mapping in a Banach space, and (2) its fixed point is $V^\star$. [4 marks]

Solution.
a. Even if the pseudocode uses V as a "running" variable that is constantly being updated, it will help us in our upcoming argument to denote by V^i the value vector after i iterations, $i \geq 0$. The initial value vector is $V^0 = \mathbf{0}$. In CV, the Bellman-type updates to each state in V^{i+1} depend entirely on V^i. On the other hand, in QV, updates for subsequent states use the "most recent" values of the previous states (meaning from V^{i+1} itself). Of course, all the information needed to populate V^{i+1} is still present in V^i even under QV, only the process of going from V^i to V^{i+1} can no longer be performed by an independent update for each state. Yet, the process of obtaining V^{i+1} from V^i can still be written down in terms of an operation $V^{i+1} = B(V^i)$, where the operator $B : (S \to \mathbb{R}) \to (S \to \mathbb{R})$ is defined as below:

$$B(X)(1) \stackrel{\text{def}}{=} \max_{a \in A} \sum_{1 \leq s' \leq n} T(s, a, s')\{R(s, a, s') + \gamma X(s')\},$$

and for $2 \leq s \leq n$,

$$B(X)(s) \stackrel{\text{def}}{=} \max_{a \in A} \left(\sum_{1 \leq s' < s} T(s, a, s')\{R(s, a, s') + \gamma B(X)(s')\} + \sum_{s \leq s' \leq n} T(s, a, s')\{R(s, a, s') + \gamma X(s')\} \right).$$

Notice that $B(X)(1) = B^\star(X)(1)$. Also observe that B depends on the sequencing of the states: specifically that $B(X)(s)$ depends on $B(X)(s')$ for $1 \leq s' < s \leq n$.

b. To show that B is a contraction mapping, we use the usual $(\mathbb{R}^n, \|\cdot\|_\infty)$ Banach space. Take arbitrary $X, Y \in \mathbb{R}^n$. We have

$$|B(X)(1) - B(Y)(1)| = |B^\star(X)(1) - B^\star(Y)(1)| \leq \|B^\star(X) - B^\star(Y)\|_\infty \leq \gamma \|X - Y\|_\infty.$$

We now use induction, assuming that for $1 \leq s' < s \leq n$, $|B(X)(s) - B(Y)(s)| \leq \gamma \|X - Y\|_\infty$. Our steps below for state s are to first apply the definition of $B^\star$, use the triangle inequality ($|a + b| \leq |a| + |b|$), then apply the fact that $\gamma < 1$.

$$\begin{aligned}
|B(X)(s) - B(Y)(s)| &\leq \max_a \left| \gamma \sum_{1 \leq s' < s} T(s,a,s')\{B(X)(s') - B(Y)(s')\} + \gamma \sum_{s \leq s' \leq n} T(s,a,s')\{X(s') - Y(s')\} \right| \\
&\leq \gamma \max_a \left(\sum_{1 \leq s' < s} T(s,a,s') |B(X)(s') - B(Y)(s')| + \sum_{s \leq s' \leq n} T(s,a,s') |X(s') - Y(s')| \right) \\
&\leq \gamma \max_a \left(\sum_{1 \leq s' < s} T(s,a,s') (\gamma \|X - Y\|_\infty) + \sum_{s \leq s' \leq n} T(s,a,s') \|X - Y\|_\infty \right) \\
&\leq \gamma \max_a \left(\sum_{1 \leq s' < s} T(s,a,s') \|X - Y\|_\infty + \sum_{s \leq s' \leq n} T(s,a,s') \|X - Y\|_\infty \right) \\
&= \gamma \|X - Y\|_\infty.
\end{aligned}$$

The proof that $V^\star$ is the fixed point of B is also straightforward. We already know that $B(V^\star)(1) = B^\star(V^\star)(1) = V^\star(1)$. For $2 \leq s \leq n$, we have

$$\begin{aligned}
B(V^\star)(s) &= \max_{a \in A} \left(\sum_{1 \leq s' < s} T(s,a,s')\{R(s,a,s') + \gamma B(V^\star)(s')\} + \sum_{s \leq s' \leq n} T(s,a,s')\{R(s,a,s') + \gamma V^\star(s')\} \right) \\
&= \max_{a \in A} \left(\sum_{1 \leq s' < s} T(s,a,s')\{R(s,a,s') + \gamma V^\star(s')\} + \sum_{s \leq s' \leq n} T(s,a,s')\{R(s,a,s') + \gamma V^\star(s')\} \right) \\
&= V^\star(s).
\end{aligned}$$

Week 4

Question. This question is about the probability of transitioning between the states of an MDP over an extended period of time. Taking notations as usual, assume that you are given

- an MDP (S, A, T, R, γ),
- a start state $s_{\text{start}} \in S$,
- a policy $\pi : S \to A$,
- a non-negative integer $t \geq 0$ specifying the number of time steps elapsed (the same as the number of actions taken), and
- an arbitrary state $s_{\text{finish}} \in S$.

For $s \in S$, $t \geq 0$, let $X[t][s]$ denote the probability that the agent is in state s at time t, assuming it is in s_{start} at $t = 0$, and it takes actions according to π at every step. By initialisation, we have

$$X[0][s] = \begin{cases} 1 & s = s_{\text{start}}, \\ 0 & \text{otherwise.} \end{cases}$$

Provide pseudocode to compute $X[t][s_{\text{finish}}]$. You can refer to any subset of the input parameters—S, A, T, R, γ, s_{start}, π, t, s_{finish}—in your pseudocode, using notations as introduced in class (such as $T(s, a, s')$ to denote a transition probability and $\pi(s)$ to denote an action). To obtain full marks, you must provide an algorithm whose running time scales at most polynomially in t. [3 marks]

Solution. For $i \geq 0$, the agent is in state s at time $i + 1$ if and only if it made a transition into s from the state s' in which it was at time i. Hence we obtain a recursive relationship between $X[i + 1][\cdot]$ and $X[i][\cdot]$, which involves the transition probabilities under π. This recurrence can be used to compute $X[t][s_{\text{finish}}]$. Pseudocode is provided below.

```
For s ∈ S:
    X[0][s] ← 0.
X[0][s_start] ← 1.

For i = 0, 1, ..., t − 1:
    For s ∈ S:
        X[i + 1][s] ← Σ_{s'∈S} X[i][s'] T(s', π(s'), s).
Return X[t][s_finish].
```

The sum in the recurrence can be performed using a for-loop or a vector dot product. Notice that $O(|S|^2)$ arithmetic operations are performed to obtain $X[i+1]$ from $X[i]$, and there are t such iterations in total. There is no dependence on $|A|$.

Can you think of a way to reduce the dependence on t from linear to *logarithmic*? The answer lies in writing down $X[t]$ as the product of a t-step transition *matrix* (of dimension $|S| \times |S|$) and $X[0][\cdot]$. This matrix is itself the t-th power of a single-step transition matrix, and can be computed in $O(\log(t + 1))$ matrix multiplications. The dependence on $|S|$ remains polynomial.

Week 3

Question. You are familiar with the UCB algorithm applied to Bernoulli bandits. The algorithm selects an arm to pull by being greedy with respect to the arms' upper confidence bounds. Consider arm a that has been pulled $u_a^t \geq 1$ time(s) out of a total of $t \geq 1$ pull(s), and has empirical mean $\hat{p}_a^t$. The upper confidence bound for this arm is given by

$$\text{ucb}_a^t = \hat{p}_a^t + \sqrt{\frac{1}{2u_a^t} \ln\left(\frac{1}{\delta(t)}\right)},$$

where the common choice is to set $\delta(t) = \frac{1}{t^4}$. The upper confidence bound used by KL-UCB is

$$\text{ucb-kl}_a^t = \text{the solution } q \in [\hat{p}_a^t, 1] \text{ that satisfies } u_a^t \text{KL}(\hat{p}_a^t, q) = \ln\left(\frac{1}{\delta'(t)}\right),$$

wherein we commonly take $\delta'(t) = \frac{1}{t \cdot (\ln t)^c}$ for some fixed $c \geq 3$. Recall that for $x, y \in [0,1]$, $\text{KL}(x,y)$ denotes the KL-divergence between Bernoulli distributions with means x and y, respectively.

a. Show that if $0 < \delta(t) \leq \delta'(t)$, then regardless of the number of pulls u_a^t and empirical mean $\hat{p}_a^t$, we are guaranteed that $\text{ucb-kl}_a^t \leq \text{ucb}_a^t$. You can use the well-known Pinsker's Inequality, which states that for $x, y \in [0,1]$, $\text{KL}(x,y) \geq 2(x-y)^2$. [2 marks]

b. The intuition behind KL-UCB incurring lower regret than UCB is that it uses a "tighter" upper confidence bound, as established in part a. Extending this logic, suppose we propose an even tighter quantity

$$\text{ucb-proposed}_a^t = \frac{1}{2}\left(\hat{p}_a^t + \text{ucb-kl}_a^t\right),$$

which clearly satisfies $\text{ucb-proposed}_a^t \leq \text{ucb-kl}_a^t$. May we expect an algorithm that is greedy with respect to ucb-proposed to incur even lower regret than ucb-kl_a^t? Explain. [1 mark]

Solution.

a. By definition, $u_a^t \text{KL}(\hat{p}_a^t, \text{ucb-kl}_a^t) = \ln\left(\frac{1}{\delta'(t)}\right)$. Applying Pinsker's Inequality to the LHS and and the relation between δ and δ' to the RHS, we get $2u_a^t(\text{ucb-kl}_a^t - \hat{p}_a^t)^2 \leq \ln\left(\frac{1}{\delta(t)}\right)$, which, in turn, can be rearranged to obtain

$$\text{ucb-kl}_a^t \leq \hat{p}_a^t + \sqrt{\frac{1}{2u_a^t} \ln \frac{1}{\delta(t)}} = \text{ucb}_a^t.$$

b. Note that ucb_a^t and ucb-kl_a^t are both genuine upper confidence bounds: with probability $\delta(t)$ or $\delta'(t)$, respectively, there is a guarantee that the true mean does not exceed them. And we have formal proofs that for mistake probability $\delta(t)$ or $\delta'(t)$, the corresponding algorithm achieves logarithmic regret. On the other hand, it is not clear that ucb-proposed will be an upper bound on the mean with sufficiently high probability. It runs the risk of not exploring at a sufficient rate. An ideal upper confidence bound would be one that achieves the required "mistake probability", while still being as tight as possible.

Another perspective on the question would be that since KL-UCB is proven to be asymptotically optimal, it is not possible for any other algorithm to improve upon it substantively. However, note that there remains room for an algorithm to always achieve lower regret than KL-UCB, but only, say, by at most a constant amount. Such an occurrence would not contradict known theory.

Week 2

Question. Since the UCB algorithm achieves logarithmic regret on every bandit instance, we may infer that it satisfies the GLIE conditions. In this question, you are to argue from first principles that indeed UCB performs an infinite amount of exploration. To simplify our argument, we only consider a 2-armed bandit instance with arms 1 and 2. Suppose that the algorithm is (1) initialised by pulling each arm once, and (2) thereafter it is greedy with respect to the arms' upper confidence bounds at each time step, (3) breaking ties uniformly at random.

Adopting the usual notation, let u_a^t and $\hat{p}_a^t$ denote the number of pulls and the empirical mean of arm $a \in \{1, 2\}$ after $t \geq 2$ pulls (which ensures that the empirical means are well-defined). We consider an arbitrary t-length history h, summarised by t, u_1^t, $\hat{p}_1^t$, u_2^t, $\hat{p}_2^t$. We contemplate: *is it possible that one of the arms will never get pulled after encountering h?* Your task is to show that on the contrary, there exists a finite integer T (which can be defined in terms of t, u_1^t, $\hat{p}_1^t$, u_2^t, $\hat{p}_2^t$ or some subset of them) such that the T pulls following h are *guaranteed* to have at least one pull of each arm. It is okay of you unable to work out an explicit formula for T, but are still able to formally argue for its existence. Support your claims with rigorous justification, rather than appealing to "intuition" and informal observations. [4 marks]

Solution. Suppose that for some $x > 1$, the x pulls following h are all of the same arm, which, without loss of generality, we may take as arm 1. Then we have

$$\begin{aligned} ucb_2^{t+x} - ucb_1^{t+x} &= \hat{p}_2^{t+x} + \sqrt{\frac{2\ln(t+x)}{u_2^t}} - \hat{p}_1^{t+x} - \sqrt{\frac{2\ln(t+x)}{u_1^t + x}} \\ &\geq 0 + \sqrt{\frac{2\ln(t+x)}{t}} - 1 - \sqrt{\frac{2\ln(t+x)}{x}} \\ &= -1 + \sqrt{\frac{2\ln(t+x)}{t}} \left(1 - \sqrt{\frac{t}{x}}\right). \end{aligned}$$

If we choose any $x \geq e^t + 16$, we have $\ln(t+x) > t$ and $\sqrt{t/x} \leq \sqrt{2/(e^2+16)} < 0.2925$, using the fact that $\sqrt{\frac{t}{e^t+16}}$ is a decreasing function of t, which is maximised in our domain at $t = 2$. Consequently we get

$$ucb_2^{t+x} - ucb_1^{t+x} > -1 + \sqrt{2}(1 - 0.2925) > 0.$$

Thus, even if all x pulls have been of arm 1, it is clear that the next pull $(t + x + 1)$ must be of arm 2. For the choice of $T = \lceil e^t \rceil + 17$, we have established that the T pulls following h cannot all be of the same arm.

Week 1

Question. Consider the family of n-armed bandit instances, $n \geq 2$, in which each arm $a \in \{1, 2, \ldots, n\}$ generates a 1-reward with probability p_a and a 0-reward with probability $1 - p_a$. Thus, each instance of the family is fixed by a vector $(p_1, p_2, \ldots, p_n)$, where $p_a \in [0, 1]$ for $a \in \{1, 2, \ldots, n\}$.

A round-robin algorithm undertakes $m \geq 2$ passes over the set of arms; the sequence of pulls $1, 2, \ldots, n$ is repeated m times. For each arm $a \in \{1, 2, \ldots, n\}$, let s_a denote the number of 1-rewards (interpreted as "successes") from its m pulls, and let f_a denote the number of 0-rewards (interpreted as "failures") from its m pulls (hence $s_a + f_a = m$).

a. For a fixed bandit instance $(p_1, p_2, \ldots, p_n)$, what is the probability that $s_1 = s_2 = \cdots = s_n$? Give your answer in terms of $p_1, p_2, \ldots, p_n$, and m. [2 marks]

b. Denote the total number of successes after the m passes $S = s_1 + s_2 + \cdots + s_n$. What are the mean and variance of S? Again, your answer must be in terms of $p_1, p_2, \ldots, p_n$, and m. [2 marks]

It will help to view the reward given by each pull as a random variable, noting that it is independent of the $(nm - 1)$ others. This view can facilitate an easy computation of the variance of S in part b—in your answer, be sure to explain why.

Solution.

a. Each arm a is pulled m times. The probability that it gets s_a successes and f_a failures for $0 \leq s_a \leq m$, $s_a + f_a = m$, is $\binom{m}{s_a}(p_a)^{s_a}(1 - p_a)^{f_a}$. For any fixed number of successes $s \in \{0, 1, \ldots, m\}$, the probability that all n arms get s successes is

$$\prod_{a \in \{1,2,\ldots,n\}} \binom{m}{s} (p_a)^s (1 - p_a)^{m-s}.$$

The required probability takes into account all possible values of s, and is thus

$$\sum_{s=0}^{m} \prod_{a \in \{1,2,\ldots,n\}} \binom{m}{s} (p_a)^s (1 - p_a)^{m-s}.$$

b. S is seen to be the sum of nm Bernoulli variables $X_{a,l}$ for arm $a \in \{1, 2, \ldots, n\}$ and pass $l \in \{1, 2, \ldots m\}$. The mean of $X_{a,l}$ is p_a, and its variance is $p_a(1 - p_a)$. We use

$$\mathbb{E}[S] = \sum_{a=1}^{n} \sum_{l=1}^{m} \mathbb{E}[X_{a,l}] = m \sum_{a=1}^{n} p_a.$$

Since the variables are independent, we also have

$$\mathrm{Var}[S] = \sum_{a=1}^{n} \sum_{l=1}^{m} \mathrm{Var}[X_{a,l}] = m \sum_{a=1}^{n} p_a(1 - p_a).$$

Note that if random variables X and Y are *not* independent, it is not necessary that they satisfy $\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]$. In typical bandit algorithms (such as ϵ-greedy sampling), the *arm* that is pulled at some fixed time step could itself be random, disallowing the decomposition of S into $\sum_{a=1}^{n} \sum_{l=1}^{m} X_{a,l}$, which makes our variance-calculation convenient.