CS 747 (Autumn 2022) Mid-semester Examination

Instructor: Shivaram Kalyanakrishnan

6.30 p.m. – 8.30 p.m., September 15, 2022, LA 201 and LA 202

Note. This exam has **4** 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 the ϵG1 algorithm described in class, when applied to an n-armed bandit, $n \geq 2$. The arms yield 0 and 1 rewards, with arm $a \in \{1, 2, \ldots, n\}$ having mean reward p_a. Given a horizon of T pulls (assume T is sufficiently large), the algorithm first pulls each arm $\lceil \frac{\epsilon T}{n} \rceil$ times (the version in class gives each arm the same expected number of pulls, but here the actual number is identical, so subsequent analysis is simplified). Upon completing this exploratory phase, the algorithm identifies an arm a^{best} with the highest empirical mean, and for the remaining $T - n\lceil \frac{\epsilon T}{n} \rceil$ pulls, repeatedly samples a^{best} (not bothering to check if its empirical mean now falls below any other arm's.) The parameter $\epsilon \in (0, 1)$ controls the amount of exploration.

Let $I = (p_1, p_2, \ldots, p_n)$ denote the bandit instance on which ϵG1 is executed for horizon T. Show that there exists $\epsilon(I, T)$—that is, a setting of ϵ that depends on the bandit instance and the horizon—such that if run with $\epsilon = \epsilon(I, T)$, the expected cumulative regret of the algorithm is at most $C(I) \cdot \ln(T)$, where $C(I)$ is a quantity that depends only on I.

We have shown such a result in class for the UCB algorithm. Whereas UCB did not require any parameters to be tuned so as to achieve this result, this question asks you to furnish the argument that ϵG1 can also achieve logarithmic regret *provided it is tuned appropriately.* [5 marks]

Question 2. Consider the Thompson Sampling algorithm presented in class for bandits whose arms yield 0 and 1 rewards. Suppose the algorithm is initialised (as usual) with a uniform prior as the belief distribution for each arm. Also suppose we run it on a 2-armed bandit instance, in which the arms 1 and 2 have mean rewards p_1 and p_2, respectively. What is the probability that both arms get pulled within the first two pulls? In other words, what is the probability that the sequence of pulls is either (arm 1, arm 2) or (arm 2, arm 1)?

For your calculations, you can use the following formula for the pdf of the Beta distribution with integer parameters $\alpha, \beta \geq 1$, evaluated at $x \in [0, 1]$.

$$\text{Beta}(x; \alpha, \beta) = \frac{(\alpha + \beta - 1)!}{(\alpha - 1)!(\beta - 1)!} x^{\alpha - 1}(1 - x)^{\beta - 1}.$$

Take $0! = 0^0 = 1$ in case you encounter these quantities. Your answer should be in terms of p_1 and p_2. [5 marks]

Question 3. X, Y, and Z are nodes in a delivery network, which is shown in the figure below. X is the source node, and has a set of $N \geq 2$ identical packets to be transported to the target node Z. Y is an intermediate node through which packets can be shipped from X to Z, over and above the direct XZ link.

$$\begin{tikzcd}
 & Y \arrow[dr, "p_{YZ}"] & \\
X \arrow[ur, "p_{XY}"] \arrow[rr, "p_{XZ}"'] & & Z
\end{tikzcd}$$

Shipments are undertaken on a daily basis, at 10.00 a.m., from each node. At most one packet can be put on any link on any given day. When a packet is sent from a node on a link, it *succeeds* with a fixed probability (denoted p_{XY}, p_{YZ}, p_{XZ} for the respective links). If shipped on day i, a successful delivery reaches the destination by 9.00 a.m. the next day, and hence is available to be shipped onward from the destination the same day. However, if the delivery fails, the packet remains "in transit" on the link; its delivery will again be attempted the next day, and again it succeeds with the same link-success-probability. Thus, for example, a shipment from X to Y sent out on day i will reach Y on day $i + 1$ with probability p_{XY}, on day $i + 2$ with probability $(1 - p_{XY})p_{XY}$, on day $i + 3$ with probability $(1 - p_{XY})^2 p_{XY}$, and so on. When a shipment is stuck on a link, a new shipment cannot be sent on the same link. However, there is sufficient capacity at each node to hold packets. Thus, a shipment can be sent from the source to the destination of a link so long as the link is free, no matter how many packets are already present at the destination.

The shipping company running this network has set up infrastructure so that the location of each packet (which is one of the nodes or the links) is known to the control room at every instant of time. The control room must instruct X and Y at 9.30 a.m. every day as to what they must do: that is, whether or not to send a shipment out on each of their outgoing links. As you can imagine, the instructions provided by the control room will determine the overall time taken to transfer all N packets from X to Z.

Your task is to help the control room by formulating an MDP based on the application described, so that an optimal policy for the MDP will readily yield the actions to be taken at X and Y during their daily operation. Concretely, the aim is to minimise the expected number of days to transfer all N packets from X to Z. All links are free to begin. Write down your formulation, providing sufficient explanation, and stating any assumptions you might make. [5 marks]

Question 4. You are given an MDP $M = (S, A, T, R, \gamma)$, with notation and conventions as usual. You are also given a function $V : S \to \mathbb{R}$. Your task is to come up with an algorithm to answer the following question:

> *"Does there exist a policy π for which the value function V^{π} (on M) is identical to V (that is, for all $s \in S$, $V^{\pi}(s) = V(s)$)?"*

It is not necessary that π be a deterministic policy; if there is a stochastic policy π such that $V^{\pi} = V$, your algorithm must give "Yes" as output (and otherwise "No"). You do not have to consider time-varying or history-dependent policies: consider only the set of Markovian, stochastic policies (which, of course, includes all Markovian, deterministic policies).

You are *encouraged* to use linear programming as a part of your algorithm. In other words, your algorithm can formulate a linear program based on its inputs M and V, and assume that a blackbox solver will compute the solution to the linear program. If you only have to check whether your linear program has a feasible solution (it has no objective function to maximise), specify a "dummy" objective function such as a constant. If $\mathcal{F}$ is the set of all feasible solutions that maximise the objective function, you can assume that the the solver will return an arbitrary element of $\mathcal{F}$ if $\mathcal{F}$ is non-empty, and declare that there is "No feasible solution" if $\mathcal{F}$ is empty.

You are also welcome to present approaches that do not use linear programming as a subroutine; you will receive full credit for any solution that is correct. In any case, after you describe your algorithm, provide a few lines arguing for its correctness. [5 marks]

Solutions

1. If the means $p_1, p_2, \dots, p_n$ are all identical, there is no regret incurred. Let us proceed by assuming that the arms' means are not all equal, and let $\Delta = p_{\text{first}} - p_{\text{second}}$, where p_{first} is the largest among the means, and p_{second} the second-largest.

Notice that R_{explore}, the regret from the initial exploratory phase of $n\lceil \frac{\epsilon T}{n} \rceil$ pulls, is *at least* $\frac{\Delta \epsilon T}{n}$, since each suboptimal arm gets pulled at least $\frac{\epsilon T}{n}$ times in this phase. R_{explore} can only scale as $O(\log(T))$ if ϵ is in the order of $O(\frac{\log(T)}{T})$. Let us assume $\epsilon = c\frac{\ln T}{T}$. Then we have $R_{\text{explore}} \leq \Delta(c\ln(T) + n)$.

The "exploit" phase comprises the $T - n\lceil \frac{\epsilon T}{n} \rceil$ pulls given to a^{best} after it has been identified. Since we have assumed $\epsilon = c\frac{\ln T}{T}$, we conclude that there are $\Omega(T)$ pulls of a^{best} in the exploit phase. Naturally R_{exploit}, the regret from this phase, can be $O(\log T)$ only if the probability that a^{best} is not optimal is sufficiently small. Indeed let δ denote the probability that a^{best} is not optimal. We can be sure that a^{best} is indeed optimal if after the explore phase, the empirical mean of some optimal arm did not drop $\frac{\Delta}{2}$ or more below its true mean, and the empirical mean of every suboptimal arm did not come out $\frac{\Delta}{2}$ or more above its true mean. Since each arm is pulled $\lceil \frac{\epsilon T}{n} \rceil$ times, we have

$$\delta \leq ne^{-2\lceil \frac{\epsilon T}{n} \rceil (\frac{\Delta}{2})^2} \leq ne^{-2\frac{\epsilon T}{n}(\frac{\Delta}{2})^2} = ne^{-2\frac{c\ln T}{n}(\frac{\Delta}{2})^2} = n\left(\frac{1}{T}\right)^{\frac{c\Delta^2}{2n}}.$$

Since $R_{\text{exploit}} \leq \delta \Delta T$, we have $R_{\text{exploit}} \leq \Delta n T^{1-\frac{c\Delta^2}{2n}}$, which is at most Δn (a quantity independent of T) if $c \geq \frac{2n}{\Delta^2}$. In summary, by running $\epsilon G1$ with $\epsilon = \frac{2n}{\Delta^2}\frac{\ln T}{T}$, we observe that the regret is

$$R_{\text{exploit}} + R_{\text{explore}} \leq \Delta(c\ln(T) + n) + \Delta n = O(logT).$$

2. Let $h = (a^0, r^0, a^1)$ denote the action-reward-action sequence that begins the agent's interaction with the bandit instance. We are to calculate the probability that h is from the set $\{(1,0,2), (1,1,2), (2,0,1)(2,1,1)\}$. We have:

$$\begin{aligned}
\mathbb{P}\{h = (1,0,2)\} &= \frac{1}{2} \cdot (1 - p_1) \cdot \int_{x=0}^{1} \int_{y=x}^{1} \text{Beta}(x;1,2)\text{Beta}(y;1,1)dydx \\
&= \frac{1-p_1}{2} \int_{x=0}^{1} \frac{2!}{1!0!}(1-x)(1-x)dx = \frac{1-p_1}{3}, \text{ and} \\
\mathbb{P}\{h = (1,1,2)\} &= \frac{1}{2} \cdot p_1 \cdot \int_{x=0}^{1} \int_{y=x}^{1} \text{Beta}(x;2,1)\text{Beta}(y;1,1)dydx \\
&= \frac{p_1}{2} \int_{x=0}^{1} \frac{2!}{0!1!} x(1-x)dx = \frac{p_1}{6}.
\end{aligned}$$

By symmetry, we have $\mathbb{P}\{h = (2,0,1)\} = \frac{1-p_2}{3}$, and $\mathbb{P}\{h = (2,1,1)\} = \frac{p_2}{6}$. Consequently the required probability is

$$\mathbb{P}\{h = (1,0,2)\} + \mathbb{P}\{h = (1,1,2)\} + \mathbb{P}\{h = (2,0,1)\} + \mathbb{P}\{h = (2,1,1)\} = \frac{2}{3} - \frac{(p_1 + p_2)}{6}.$$

3. From the description of the application, we observe that the consequences of actions are completely determined by the number of packets present at each node and link. Since there is no decision to make at Z, we can ignore the number of packets present there, and treat state as $(n_X, n_Y, n_{XY}, n_{YZ}, n_{XZ})$, where n_W denotes the number of packets in node or link W. This formulation results in $\Theta(N^2)$ states, since there can be $\Theta(N)$packets at each node and 0 or 1 at each link.

As specified, the decision to be taken on each day is whether to send a packet from X and Y on each of their outgoing links. Consequently we can view action as a 3-bit string conveying whether a packet is to be put on XY (or not), on XZ (or not), and on YZ (or not). Hence there are 8 actions (which can possibly be pruned by reasoning that some actions are inapplicable or suboptimal from some states, but we do not worry about efficiency for now). The main point to note about the states and actions is that they apply to the entire network, not to individual nodes/links. This conjunctive modeling is necessary to achieve efficiency in operation.

The transition function naturally factorises into rules that operate on each node-link and link-node interface. Consider state $(n_X, n_Y, n_{XY}, n_{YZ}, n_{XZ})$ and action (a_{XY}, a_{XZ}, a_{YZ}). In general, multiple possible next states of the form $(n'_X, n'_Y, n'_{XY}, n'_{YZ}, n'_{XZ})$ may be reachable with non-zero probability. We specify a couple of illustrative transitions. If $n_X > 0, n_{XY} = 0$ and $a_{XY} = 1$, then we go with probability p_{XY} to a state with $n'_Y = n_Y + 1$; with the remaining probability we reach a state with $n'_{XY} = 1, n'_Y = n_Y$ (provided $a_{YZ} = 0$). If $n_{XY} = 1$, then we have $n_{XY} = 0, n'_Y = n_Y + 1$ with probability p_{XY}, and status quo—$n_{XY} = 1, n'_Y = n_Y$—with probability $1 - p_{XY}$ (provided $a_{YZ} = 0$). If $n_X > 1, n_{XY} = 0, n_{XZ} = 0$ and $a_{XY} = 1, a_{XZ} = 1$, then we have $n'_X = n_X - 2$ with probability $p_{XY}p_{XZ}$, $n'_X = n_X$ with probability $(1 - p_{XY})(1 - p_{XZ})$, and $n'_X = n_X - 1$ with the remaining probability. In each case n_{XY}, n_{XZ}, and n_Y are modified suitably.

In order to minimise the expected number of days to complete the transaction, we give a reward of -1 on each day, leaving the task undiscounted. The task is guaranteed to terminate provided $p_{XZ} > 0$ or $p_{XY}p_{YZ} > 0$, and packets are guaranteed to be pushed out from X and Y (which is easily implemented). We could also use the reward function to eliminate illegal actions. Thus, for example, setting $a_{XY} = 1$ when $n_{XY} = 1$ could be given a reward of $-\infty$.

4. Consider the following linear program, with variables $\pi(s, a)$ for $s \in S, a \in A$, and a dummy objective function.

$$\begin{aligned}
&\text{Maximise } 1 \\
&\text{subject to} \\
&\pi(s,a) \geq 0 \text{ for } s \in S, a \in A, && C1 \\
&\sum_{a \in A} \pi(s,a) = 1 \text{ for } s \in S, && C2 \\
&V(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma V(s')) \text{ for } s \in S. && C3
\end{aligned}$$

The equality constraints $C2$ and $C3$, of the form $\alpha = \beta$, can be replaced by inequalities $\alpha \leq \beta$ and $\beta \leq \alpha$ if needed. Our proposed algorithm returns "Yes" if this linear program has a feasible solution, and "No" otherwise.

Now, if indeed there is some policy π whose value function is V, it is clear that π will satisfy both the policy constraints $C1$ and $C2$, and as the Bellman equations encapsulated in $C3$. Hence the linear program will have a feasible solution, and our answer will be "Yes". Moreover, if π is a feasible solution returned by the linear program solver, it is guaranteed to satisfy the policy constraints C1 and C2, and satisfy C3, which would imply that V is indeed the value function of π. Hence, we will answer "Yes" if and only if there exists a policy π for which $V^{\pi} = V$.

It is a good idea to reuse general-purpose tools such a linear programming solvers, but in this particular case, a closer look at the LP suggests that its feasibility is much simpler to establish. Notice that for any fixed $s \in S$, the $|A|$ policy variables $\pi(s, a)$ do not occur in C3 constraints other than the one for s itself (with $V(s)$ on the LHS). For $s \in S, a \in A$, define $Q(s,a) = \sum_{s' \in S} T(s,a,s')(R(s,a,s' + \gamma V(s')))$. The LP is merely posing the question: for each $s \in S$, can $V(s)$ be expressed as a convex combination of the $Q(s, a)$ values, $a \in A$? In turn, the answer is affirmative if and only if for each $s \in S$,

$$\min_{a \in A} Q(s,a) \leq V(s) \leq \max_{a \in A} Q(s,a),$$

which can be easily verified.