CS 170 Homework 9

Due **4/4/2022, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 How to Gamble With Little Regret

Suppose that you are gambling at a casino. Every day you play at a slot machine, and your goal is to minimize your losses. We model this as the experts problem. Every day you must take the advice of one of n experts (i.e. play at a slot machine). At the end of each day t, if you take advice from expert i, the advice costs you some c_i^t in $[0, 1]$. You want to minimize the regret R, defined as:

$$R = \frac{1}{T}\left(\sum_{t=1}^{T} c_{i(t)}^t - \min_{1\le i\le n} \sum_{t=1}^{T} c_i^t\right)$$

where $i(t)$ is the expert you choose on day t. Notice that in this definition, you are comparing your losses to the best expert, rather than the best overall strategy.

Your strategy will be probabilities where p_i^t denotes the probability with which you choose expert i on day t. Assume an all powerful adversary (i.e. the casino) can look at your strategy ahead of time and decide the costs associated with each expert on each day. Let C denote the set of costs for all experts and all days. Compute $\max_C(\mathbb{E}[R])$, or the maximum possible (expected) regret that the adversary can guarantee, for each of the following strategies.

(a) Any deterministic strategy, i.e. for each t, there exists some i such that $p_i^t = 1$.

(b) Always choose an expert according to some fixed probability distribution at every time step. That is, fix some $p_1, \ldots, p_n$, and for all t, set $p_i^t = p_i$.

What distribution minimizes the regret of this strategy? In other words, what is $\text{argmin}_{p_1,\ldots,p_n} \max_C(\mathbb{E}[R])$?

This analysis should conclude that a good strategy for the problem must necessarily be randomized and adaptive.

Solution:

(a) $\frac{n-1}{n}$. Consider the case where the cost of the chosen expert is always 1, and the cost of each other expert is 0. Let k be the least-frequently chosen expert, and let m_k be the number of times that expert is chosen. This will result in a regret of $\frac{1}{T}(T - m_k)$

Since the best expert is the one that is chosen least often, the best strategy will try to maximize the number of times we choose the expert that is chosen least often. This

means we want to choose all the experts equally many times, so expert k is chosen in at most T/n of the rounds. Therefore, $m_k \leq \frac{T}{n}$, thus the regret is at least $\frac{1}{T}(T - \frac{T}{n}) = \frac{n-1}{n}$. (Note here that even if our strategy is adaptive, i.e. it chooses an expert on day i based on the losses from days 1 to $i-1$, rather than committing to an expert for day i before seeing the loss for day 1, it still can't achieve regret better than $\frac{n-1}{n}$.)

(b) $(1 - \min_i p_i)$. Like in part (a), the distribution is fixed across all days, so we know ahead of time which expert will be chosen least often in expectation. Let $k = \text{argmin}_i p_i$ be the expert with least cost. Let $c_k^t = 0$ for all t, and let $c_i^t = 1$ for all $i \neq k$ and for all t. This way, $\mathbb{E}\left[\sum_{t=1}^T c_{i(t)}^t\right] = T\left(\sum_{i \neq k} p_i \cdot 1 + p_k \cdot 0\right) = T(1 - p_k)$. We also have $\min_i \sum_{t=1}^T c_i^t = 0$, so we end up with an expected regret of $\frac{1}{T}(T(1-p_k) - 0) = 1 - p_k$.

To minimize the expectation of R is the same as maximizing $\min_i p_i$, which is achieved by the uniform distribution. This gives us regret $\frac{n-1}{n}$ (this is the same worst case regret as in part (b)).

3 Weighted Rock-Paper-Scissors

You and your friend used to play rock-paper-scissors, and have the loser pay the winner 1 dollar. However, you then learned in CS170 that the best strategy is to pick each move uniformly at random, which took all the fun out of the game.

Your friend, trying to make the game interesting again, suggests playing the following variant: If you win by beating rock with paper, you get 4 dollars from your opponent. If you win by beating scissors with rock, you get 2 dollars. If you win by beating paper with scissors, you get 1 dollar.

(a) Draw the payoff matrix for this game.

(b) Write a linear program to find the optimal strategy.

Solution:

(a)

		Your Friend: rock	paper	scissors
You:	rock	0	-4	2
	paper	4	0	-1
	scissors	-2	1	0

(b) Let r, p, s be the probabilities that you play rock, paper, scissors respectively. Let z stand for the expected payoff, if your opponent plays optimally as well.

$$\begin{aligned} \max \quad & z \\ 4p - 2s &\geq z && \text{(Opponent chooses rock)} \\ s - 4r &\geq z && \text{(Opponent chooses paper)} \\ 2r - p &\geq z && \text{(Opponent chooses scissors)} \\ r + p + s &= 1 \\ r, p, s &\geq 0 \end{aligned}$$

4 Domination

In this problem, we explore a concept called *dominated strategies*. Consider a zero-sum game with the following payoff matrix for the row player:

		Column: A	B	C
Row:	D	1	2	-3
	E	3	2	-2
	F	-1	-2	2

(a) If the row player plays optimally, can you find the probability that they pick D without directly solving for the optimal strategy? Justify your answer.

(Hint: How do the payoffs for the row player picking D compare to their payoffs for picking E?)

(b) Given the answer to part a, if the both players play optimally, what is the probability that the column player picks A?

(c) Given the answers to part a and b, what are both players' optimal strategies?

Note: All parts of this problem can be solved without using an LP solver or solving a system of linear equations.

Solution:

(a) 0. Regardless of what option the column player chooses, the row player always gets a higher payoff picking E than D, so any strategy that involves a non-zero probability of picking D can be improved by instead picking E.

(b) 0. We know that the row player is never going to pick D, i.e. will always pick either E or F. But in this case, picking B is always better for the column player than picking A (A is only better if the row player picks D). That is, conditioned on the row player playing optimally, B dominates A.

(c) Based on the previous two parts, we only have to consider the probabilities the row player picks E or F and the column player picks B or C. Looking at the 2-by-2 submatrix corresponding to these options, it follows that the optimal strategy for the row player is to pick E and F with probability 1/2, and similarly the column player should pick B, C with probability 1/2.

5 (Optional) Minimum Infinity-Norm Cut

In the Minimum Infinity-Norm Cut problem, you are given a connected undirected graph $G = (V, E)$ with positive edge weights w_e, and you are asked to find a cut in the graph where the largest edge in the cut is as small as possible (note that there is no notion of source or target; any cut with at least one node on each side is valid).

Show how to reduce Minimum Infinity-Norm Cut to Minimum Spanning Tree in linear time. **Give a 3-part solution.**

Hint: Minimum Spanning Tree does not require edge weights to be positive.

Solution: Algorithm: First, negate all the edge weights in G, and pass this new graph to whatever minimum spanning tree algorithm we are using; this will give us a maximum spanning tree of the original graph. Remove the smallest-weight edge in this maximum spanning tree, and return the cut induced by its removal.

Proof of Correctness: First note that we correctly find a maximum spanning tree of the graph, since $\max_{\text{tree } T} \sum_{e \in T} w_e = \min_{\text{tree } T} \sum_{e \in T} -w_e$. It is similarly easy to see that we find a minimum infinity-norm cut in the maximum spanning tree of the graph (using any

other edges from the spanning tree could not decrease the cut, since we used only the smallest possible edge).

It only remains to show that any cut of the nodes in the graph has exactly the same infinity norm in the graph overall as in the maximum spanning tree; this will prove the correctness of our algorithm (since we have already proven its correctness in the tree). To see this, we will use the cut property for maximum spanning trees (which follows immediately from the cut property for minimum spanning trees, applied to the negated graph). This property is: for any cut in the graph, its largest edge (or one of its largest edges, if the largest edge is not unique) must be contained in the maximum spanning tree. Since the infinity norm of the cut is equal to the weight of its largest edge, this means that the infinity norm of the cut in the tree is at least its infinity norm in the graph; it is also at most the infinity norm in the graph, since no edges exist in the tree which do not exist in the graph.

Runtime Analysis: Creating a new graph with every edge negated takes $O(|V| + |E|)$ time. Once we have the maximum spanning tree, the smallest-weight edge can be found with a simple $O(|E|)$ time search; once it is removed, the nodes in the two resulting components can be enumerated with a $O(|V| + |E|)$ time traversal. So overall, we have taken linear time.