Title Here

Daniel Deng

1 Study Group

(a) Auston, SID 3034554056

(b) Yes

2 How to Gamble with Little Regret

(a) Because the casino knows the moves of the player ahead of the time, the expected value of total losses for the player will be T, since the casino will set the chosen machine to always lose. In fact, the best strategy for the casino in tying to keep the expected loss of any machines low is to assign a loss of 1 to the machine to be chosen, and assign a loss of 0 to all other machines.

In order to minimize regret, the optimal strategy for the player would be then to chose a different machine each day, until all machines have been used once. This would force each machine to be used at least once every $\frac{T}{n}$ days, meaning that the "best" machine will still loss by 1 every $\frac{T}{n}$ days. Therefore, the maximum expected regret is

$$\max_C \mathbb{E}[R] = \frac{1}{T}\left(T - \frac{T}{n}\right) = 1 - \frac{1}{n}$$

(b) In order to guarantee max loss for the player, the casino would assign a loss value of 1 to all machine so that the chosen probability distribution do not matter. However, in order to minimize the loss of the best machine, the best strategy of the casino would be to have a loss value of 0 on the machine that is least likely to be chosen by the player. Given this scheme, the expected loss of the player each day is given by $(1 - \min_{1\leq i\leq n} p_i)$, and the best machine will always have a total loss of 0. Therefore, the expected regret would be

$$\max_C \mathbb{E}[R] = 1 - \min_{1\leq i\leq n} p_i$$

It is apparent that the best strategy for the player in this situation is to uniformly choose a machine each day (i.e., with probability $\frac{1}{n}$), which leads to

$$\operatorname*{argmin}_{p_1,\ldots,p_n} \max_C \mathbb{E}[R] = 1 - \frac{1}{n}$$

3 Zero-Sum Battle

(a)

$$\begin{aligned} \max z \\ -10x_1 + 4x_2 + 6x_3 \geq z \\ 3x_1 - x_2 - 9x_3 \geq z \\ 3x_1 - 3x_2 + 2x_3 \geq z \\ x_1 + x_2 + x_3 = 1 \\ x_1, x_2, x_3 \geq 0 \end{aligned}$$

Running the provided LP solver, we find that the optimal strategy for Trainer A is

$$\begin{cases} P[dragon] = 0.3346457 \\ P[steel] = 0.5629921 \\ P[rock] = 0.1023622 \end{cases}$$

and the expected payoff is -0.480315.

(b)

$$\begin{aligned} \min w \\ -10y_1 + 3y_2 + 3y_3 \leq w \\ 4y_1 - y_2 - 3y_3 \leq w \\ 6y_1 - 9y_2 + 2y_3 \leq w \\ y_1 + y_2 + y_3 = 1 \\ y_1, y_2, y_3 \geq 0 \end{aligned}$$

Running the provided LP solver, we find that the optimal strategy for Trainer B is

$$\begin{cases} P[ice] = 0.2677165 \\ P[water] = 0.3228346 \\ P[fire] = 0.4094488 \end{cases}$$

and the expected payoff is -0.480315, which is identical to that of Trainer A.

4 Minimum ∞-Norm Cut

Algorithm Description:
First, negate all edge weights in G as pre-processing. Then, find the MST of the modified graph. Finally, for post-processing, find the tree edge that has the largest weight by iterating through all the tree edges and keeping track of the current largest, and remove it; the two sets of vertices in the MST resulting from the deletion is the solution cut.

Proof of Correctness:
Based on the cut property of MST, each tree edge selected in MST is the smallest edge in their respective cut. Since the edges have negated weights, this would imply that each tree edge is actually the largest edge in their cut in the original graph. Finding the cut partition by disconnecting the tree at the largest weight is valid because that edge, by the same logic as before, represents the smallest largest edge in all cuts in the original graph.

Runtime Analysis:
For both pre- and post-processing, we iterate through the edges once, which implies $O(|E|)$ time complexity for the reduction.