

CS381 Assignment 2

Due October 2 2023 11:59 PM AoE

1. It is known that for games of finitely many players with finitely many strategies, there must be a Nash Equilibrium. As such, the existence of a Nash Equilibrium in the Tragedy of the Commons game from the last assignment is somewhat coincidental, as each player technically has infinitely many strategies (they could pick any $x_i \in (0, 1)$).

Consider the following 2 player game: each player picks some $x_i \in (0, 1)$. Whoever picked the higher number wins \$1, the lower gets nothing. Describe why there is no Nash Equilibrium in this game - even allowing for mixed strategies.

2. Consider a game that happens to have at least two Nash Equilibria, under which P_1 would use the (possibly mixed) strategies x_1 and x_2 and P_2 would use the (possibly mixed) strategies y_1 and y_2 (so P_1 doing x_1 and P_2 doing y_1 is a Nash Equilibrium, along with P_1 doing x_2 and P_2 doing y_2). Consider the strategy $\bar{x}$ for P_1 given by picking between x_1 and x_2 equally at random and the analogous strategy $\bar{y}$ for P_2 given by picking between y_1 and y_2 equally at random. Is this specification of strategies necessarily a Nash Equilibrium? If so, explain why, otherwise determine a game where the resultant strategies are not a Nash Equilibrium.
3. Determine all Nash Equilibria of the following game, and explain why there aren't any others:

	1	0	1
1	10	-10	1
0	1	10	1

4. In class we described a brute force algorithm for determining all Nash Equilibria of a game given in normal form by enumerating every possible combination of row player options and column player options, then solving systems of linear equations to determine the probabilities each player would have to use among these combinations in order to influence their adversary to be indifferent to their options. Describe why (despite being not even complete in description) this algorithm as stated cannot possibly be implemented in a runtime that's polynomial in the number of rows and column of the given matrix.
5. Regarding the experts' game, we tried a few algorithms:
 - i Pick any expert among the experts who have been wrong the least.
 - ii Pick randomly among the experts who have been wrong the least.
 - iii The multiplicative weights algorithm

Recall that $L_{\min}^t$ is the minimal number of times that any expert has been wrong by time t .

- (a) Let the number of experts $N = 3$. Determine an explicit sequence of length t on which the first above algorithm is wrong *at least* $N L_{\min}^t + N - 1$; explain your result.
- (b) Implement the multiplicative weights algorithm in your preferred language. We're going to use the algorithm to play the game of evens / odds defined in class. Recall that this is a 2 player game where each player says a number: if the sum of the numbers is even, P_1 wins. Otherwise, P_2 wins. Let's have P_1 use the multiplicative weights algorithm to play the game. To start, assume two experts: one telling P_1 to say an odd number, the other telling P_2 to say an even number.
- Pick $\eta = \frac{1}{100}$ and simulate the game for 1000 rounds where P_2 always says an odd number. How does P_1 do for the first 100 rounds? How do they do for the last 100?
 - Pick $\eta = \frac{1}{100}$ and simulate the game for 1000 rounds where P_2 says an even number with 50% probability (and an odd number otherwise). How does P_1 do - does it match your expectations? What do the weights look like between the two experts?
 - Do one of the following:
 - Letting L_A^t be the number of times P_1 loses by time t , calculate the average value of $L_A^t - L_{\min}^t$ for each $t \in \{1, \dots, 10000\}$ (take the average over some large number of experiments, say 10000) and show that it grows roughly similarly to $\frac{\sqrt{t}}{2.5}$.
 - Prove that *every* algorithm A would have that $E[L_A^t - L_{\min}^t] = \Omega(\sqrt{t})$ against P_2 's strategy. Hint: you'll almost certainly need to use [Stirling's approximation](#) at some point.
- Recall in class that we showed that $E[L_A^t] \leq (1 + \eta)L_{\min}^t + \frac{\log(N)}{\eta}$, so by choosing $\eta = \min(\sqrt{\frac{\log(N)}{t}})^1$ and noting that $L_{\min}^t \leq t$ we would have $E[L_A^t] \leq L_{\min}^t + 2\sqrt{t \log(N)}$. So A. shows that the bound we obtained for the number of times we are wrong by following the multiplicative weights algorithm is relatively tight, ie we couldn't utilize some powerful mathematics and have established an impressively better bound. Moreover, B. shows that the multiplicative weights algorithm does just about as well as any other algorithm when the number of options $N = 2$.

¹One might notice there is a bit of a question mark of how η is a constant if I'm just suddenly letting it depend on t . There are ways around this, but for now pretend this is allowed.