Problem Set 1

Due date: Oct 12, 2017 in class

1. Show that, in a normal form game with finitely many players, each with finitely many actions, the set of correlated equilibrium can be represented as a polytope (i.e., the feasible region of a linear program).

2. We mentioned in class that, for general games, computing a Nash equilibrium is believed to be compuationally hard. In this problem we develop an exponential-time algorithm for finding a Nash for a two-player game. Let $[n] = \{1, \dots, n\}$ and $[m] = \{1, \dots, m\}$ be the sets of pure strategies of the row and column players respectively, and suppose the entries of the utility matrix are all rational numbers in $[0, 1]$ each with bit complexity at most b.

   (a) For any $S \subseteq [n], T \subseteq [m]$, give a polynomial-time algorithm that, either finds a Nash equilibrium such that the mixed strategy of the row player is supported on S and that of the column player supported on T, or reports that such an equilibrium doesn't exist.

   (b) Give an algorithm that runs in time $O(2^{n+m} \operatorname{poly}(n, m, b))$ and computes a Nash equilibrium of the two-player game.
   (You may invoke polynomial-time algorithms for solving linear programs.)

3. Recall the problem of maximum weight matching problem in a bipartite graph: given a complete bipartite graph $G = (S, T)$ and nonnegative weights $(w_{st})_{s \in S, t \in T}$, one is asked to find a matching that maximizes the total weights of the edges in it. (Recall that a matching is a collection of edges such that any vertex in the graph has at most one edge in the collection that is incident to it.) You may (or should) have learned a polynomial-time algorithm for it. In this problem we investigate the performance of the greedy algorithm on it.

   The greedy algorithm ranks all edges by their weights from highest to lowest (breaking ties arbitrarily), and then considers them in this order: for every edge under consideration, if it does not create a conflict with edges already in the matching, add it to the matching; otherwise throw it.

   (a) Show that the greedy algorithm gives a matching whose total weight is at least half of the optimal.

   (b) Consider guiding the greedy algorithm with weights on the vertices: besides the edge weights, let's add vertex weight $\mathbf{v} \in \mathbb{R}^{S \cup T}$. The adjusted weight of an edge (s, t) is $w_{st} + v_s + v_t$. The adjusted greedy algorithm ranks the edges by the adjusted weight and then operates just as before. Show that for any bipartite graph and edge weights, there exists vertex weights so that the adjusted greedy algorithm finds the optimal matching with respect to the original edge weights.

   (c) (**Bonus Question):** Do part (3b) with the additional constraint that all the vertex weights are non-positive, and that the sum of all vertices' absolute values is equal to the optimal matching's total edge weights.
   (**Hint:** Think about linear program duality.)

4. We showed in class that, when players play in a game using no-regret learning algorithms, the time average of their strategy profiles constitute an ϵ-coarse correlated Nash equilibrium when they play long enough. We also showed that the existence of no-regret learning algorithm implies von Neumann's minimax theorem. This problem asks you to show that, in a two-player zero-sum game, when both players use no-regret learning algorithms, the time-average of each player's strategies over time in fact constitute an ϵ-Nash equilibrium.

5. Consider designing no-regret learning algorithms for maximizing gains (instead of minimizing costs as we did in class) in the expert setting. As we mentioned in class, it is easy to see that any deterministic algorithm incurs a worst-case regret that is linear in T. In particular, the algorithm which chooses the expert that maiximizes the total historical gains fails in this way. More formally, if the gain of expert i at time t is g_i^t, denote by G_i^t the total historical gains $\sum_{s=1}^t g_i^s$; then the deterministic algorithm that, at each time $t+1$, chooses $\arg\max_i G_i^t$, can incur linear regret over time.

   In some sense, this failure is similar to overfitting in other machine learning settings. A usual remedy to address overfitting is to add a regularizer to the objective and smooth the behavior of the algorithm. Here we consider regularizing the objective by including the entropy of the output distribution. Formally, let the algorithm choose, at time $t+1$, a probability vector $(p_1, \ldots, p_n)$, (where n is the number of experts), that maximizes

$$\sum_{i=1}^n p_i G_i^t + \lambda \sum_i p_i \log \frac{1}{p_i},$$

   for some $\lambda > 0$. Intuitively, this algorithm tries to pick the experts with good historical performances, but also would like to avoid extremely tilted distributions whose entropy is too small. Show that this algorithm is in fact just the Multiplicative Weight Update algorithm (i.e., the Hedge algorithm).