CS 170 Homework 10

Due **4/12/2023, 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 Variants on the Experts Problem

Consider the experts problem: We have n experts who predict it will either rain or shine tomorrow. At least one of the experts will always make the correct prediction. Every day we guess based on the experts if it will rain or shine tomorrow. Our goal is to make as few mistakes in our predictions as possible.

(a) Suppose we always guess the prediction of the majority of experts who have been correct so far, breaking ties arbitrarily (Note that this set is always non-empty since one expert is always correct). Show that we make at most $O(\log n)$ mistakes using this strategy.

(b) Suppose there are now k experts who are always correct instead of just one. Give a better upper bound on the number of mistakes the algorithm from the previous part makes makes.

(c) Suppose instead of one expert always being correct, we are only guaranteed that there is one expert who makes at most k mistakes. Consider the following algorithm: Let m be the least mistakes made by any expert so far. Use the prediction of the majority of experts who have made at most m mistakes so far.

Give an upper bound on the number of mistakes made by this algorithm.

3 Decision vs. Search vs. Optimization

Recall that a vertex cover is a set of vertices in a graph such that every edge is adjacent to at least one vertex in this set.

The following are three formulations of the VERTEX COVER problem:

- As a *decision problem*: Given a graph G, return TRUE if it has a vertex cover of size at most b, and FALSE otherwise.
- As a *search problem*: Given a graph G, find a vertex cover of size at most b (that is, return the actual vertices), or report that none exists.
- As an *optimization problem*: Given a graph G, find a minimum vertex cover.

At first glance, it may seem that search should be harder than decision, and that optimization should be even harder. We will show that if any one can be solved in polynomial time, so can the others. **For each of the following parts, provide a 3-part solution.**

(a) Suppose you are handed a black box that solves VERTEX COVER (DECISION) in polynomial time. Give an algorithm that solves VERTEX COVER (SEARCH) in polynomial time.

(b) Similarly, suppose we know how to solve VERTEX COVER (SEARCH) in polynomial time. Give an algorithm that solves VERTEX COVER (OPTIMIZATION) in polynomial time.

4 (OPTIONAL) Solving Linear Programs using Multiplicative Weights

In this problem, we will develop an algorithm to approximately solve linear programs using multiplicative weights. For simplicity, we will restrict our attention to a specific kind of linear programs given as follows:
Given vectors $\mathbf{a}_j$ for $j = 1, \ldots, m$ such that $|(\mathbf{a_j})_i| \leq 1$ where $i = 1, \ldots, n$, the linear program on variables $\mathbf{x} = (x_1, \ldots, x_n)$ is given by:

$$\begin{aligned} & & \max(0) \\ \text{for all} \quad & j = 1, \ldots, m : & \langle \mathbf{a}_j, \mathbf{x} \rangle \leq 0 \\ & & \sum_{i=1}^{n} x_i = 1 \\ \text{for all} \quad & i = 1, \ldots, n : & x_i \geq 0 \end{aligned}$$

Here $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the dot product between vectors, specifically

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i \cdot y_i$$

where x_i, y_i refers to the i^{th} components of the vectors $\mathbf{x}, \mathbf{y}$. Note that the objective function is trivial here because the LP formulation of MWU is a constraint satisfaction problem.

We will now use multiplicative weights update towards solving our linear program. The idea would be to execute multiplicative weights update against an appropriate adversary (sequence of losses) and let the weights of the algorithm determine the solution $\mathbf{x}$. Formally, the rough outline of our algorithm to solve the above LP is as follows:

```
For $t = 1$ to a sufficiently large number $T$
   1. Multiplicative weights update will return a current probability
      distribution $\mathbf{x}^{(t)}$
   2. If $\mathbf{x}^{(t)}$ approximately satisfies the LP, return $\mathbf{x}^{(t)}$.
   3. Adversary $\mathcal{A}$ will present a loss vector $\ell^{(t)} \in \mathbb{R}^n$.
   4. Multiplicative weights algorithm will incur a loss of $\langle \ell^{(t)}, \mathbf{x}^{(t)} \rangle$ and
      update its weights.
Return $\mathbf{x}^{(T)}$
```

We will design an algorithm that will act as the adversary $\mathcal{A}$, so that $\mathbf{x}^{(T)}$ is an approximate solution to the LP.

Recall that the multiplicative weights algorithm obeys the following regret bound:

Theorem. The multiplicative weights algorithm starts with an iterate $\mathbf{x}^{(1)} \in \mathbb{R}^n$, and then suffers a sequence of loss vectors $\ell^{(1)}, \ldots, \ell^{(T)} \in \mathbb{R}^n$ such that $\ell_i^{(t)} \in [-1, 1]$. It then produces a sequence of iterates $\mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}$ such that for some $0 < \epsilon \leq \frac{1}{2}$,

$$\sum_{t=1}^{T} \langle \ell^{(t)}, \mathbf{x}^{(t)} \rangle \leq \min_{1 \leq i \leq n} \left(\sum_{t=1}^{T} \ell_i^{(t)} \right) + 2 \left(\epsilon T + \frac{\log(n)}{\epsilon} \right)$$

(a) Let $\mathbf{p} = (p_1, \ldots, p_n)$ be a probability distribution over $\{1, \ldots, n\}$, i.e., $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$. For any vector $\mathbf{v} \in \mathbb{R}^n$ prove that,

$$\min_{1 \leq i \leq n} v_i \leq \sum_{i=1}^{n} p_i v_i$$

(In words, the minimum is smaller than the average under every distribution)

(b) We will now observe an important property of multiplicative weights. The algorithm not only has small regret against any fixed best expert, but also has small regret against any fixed probability distribution over experts.

Let $\mathbf{p}^* = (p_1^*, \ldots, p_n^*)$ be a probability distribution on $\{1, \ldots, n\}$, i.e. $p_i^* \geq 0$ for each $1 \leq i \leq n$ and $\sum_{i=1}^{n} p_i^* = 1$. Using the bounds in the regret of the multiplicative weights algorithm mentioned above, prove that

$$\sum_{t=1}^{T} \left(\langle \ell^{(t)}, \mathbf{x}^{(t)} \rangle - \langle \ell^{(t)}, \mathbf{p}^* \rangle \right) \leq 2 \left(\frac{\log(n)}{\epsilon} + \epsilon T \right)$$

(c) At the t^{th} iteration, the probability distribution $\mathbf{x}^{(t)}$ output by multiplicative weights update is a *candidate* solution to our LP. But it is likely to violate some of the constraints of the LP, namely $\langle \mathbf{a}_i, \mathbf{x}^{(t)} \rangle \leq 0$.

Describe how to implement the adversary $\mathcal{A}$ to nudge the multiplicative weights algorithm towards a feasible solution to the LP.

Hint: What loss vector should the adversary $\mathcal{A}$ pick in order to penalize the multiplicative weights algorithm the most for violations?

(d) Let us call a solution $\mathbf{x}$ to be η-approximate solution to the LP if it satisfies all the constraints within an error of η where $\eta \leq 2$, i.e.,

$$\langle \mathbf{a}_i, \mathbf{x} \rangle \leq \eta$$

.

Assume that the LP is feasible in that there exists some probability distribution $\mathbf{x}^*$ that satisfies all the constraints. Show that if multiplicative weights uses $\epsilon = \frac{\eta}{4}$ then after $T = \frac{16\log(n)}{\eta^2}$ iterations, the distribution $\mathbf{x}^{(T)}$ is an η-approximate solution.

Hint: In every iteration t, if the current distribution $\mathbf{x}^{(t)}$ violates a constraint by more than η, then it accumulates regret for not being the feasible solution $\mathbf{x}^*$.