CS 787: Advanced Algorithms — Homework 4

Out: 4/9/19 — **Due: 4/23/19**

Note: Homework should be submitted in pairs on Canvas. We will only accept PDF files.

1. **(Limited interview slots.)** Consider the following variant of the hiring problem we discussed in class. There are n candidates, each with a non-negative quality q_i drawn from a known distribution D_i. You can interview the candidates in any order. After each interview, you get to observe the quality of the candidate and must make an irrevocable decision to hire the candidate or let them go. The twist is that you may only conduct k interviews at the most.

   In this problem we will develop a constant factor competitive algorithm for this problem, but to make the task easier, we will compare the performance of the algorithm against the best online algorithm.

   (a) Consider any online algorithm and let x_i denote the probability that the algorithm interviews candidate i. Let y_{iv} denote the probability that the algorithm hires i given the candidate's quality is v. What constraints do these variables satisfy? Write a linear program expressing the maximum value such an algorithm can obtain.

   (b) Let (x^*, y^*) denote the optimal solution to your LP. Develop an algorithm for the hiring problem based on this solution that obtains a constant fraction of the value of your LP.

   Hint: Consider interviewing and hiring candidates with probability a constant factor smaller than that given by the LP solution.

2. **(Adaptive learning.)** Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.

   (a) Each expert begins with weight 1 (as before).

   (b) At every step, we predict the result of a weighted-majority vote of the experts (as before).

   (c) At every step, for every expert that makes a mistake, we penalize it by dividing its weight by 2, but only if its weight was at least $1/4$ of the average weight of experts.

   Prove that in any contiguous block of steps (e.g., the 51st step through the 77th step), the number of mistakes made by the algorithm is at most $O(m + \log n)$, where m is the number of mistakes made by the best expert in that block, and n is the total number of experts.

   (Recall that the original weighted majority algorithm only gives a guarantee over a block of steps starting at step 1.)

3. **(Sleeping experts.)** Consider a standard online prediction setting and suppose that at any time step, only a subset of the experts are available to make a prediction. Can we modify the Hedge algorithm from class to give a low regret bound in this case? More precisely, in this setting, at every time step we get to see which experts are "awake" to make a prediction. We choose one of those experts. Then we get to see the cost vector for that step. The total cost of expert i is the sum of its costs over the steps that i was awake in. Naturally, if an expert is awake for very few steps, we cannot hope to prove that our total cost over all the steps is not much larger than the expert's cost over the steps that it was awake in. So we will aim for a slightly different guarantee.

   Let T_i denote the set of steps when expert i was awake, $\text{cost}_i(\text{ALG})$ denote the expected cost of the algorithm over the steps T_i, and $\text{cost}_i(i)$ denote the cost of expert i over the steps T_i. Then, our goal is to say that $\text{cost}_i(\text{ALG}) \leq \frac{1}{1-\epsilon}\text{cost}_i(i) + O(\frac{1}{\epsilon}\log n)$, and this should hold for every expert i.

   Consider the following variant of the Hedge algorithm for this problem:

   (i) Initialize $w_{i,0} = 1$ for all i.

   (ii) At every step t, let $p_{i,t} = w_{i,t}/W_t$ be the probability of picking an awake expert i, where W_t is the sum of the weights of all the experts awake at that step (not the total weight of all the experts).

(iii) Let $c_{i,t}$ denote the cost to expert i at step t. Update weights for awake experts as follows:

$$R_{i,t} = \frac{1}{1+\epsilon}\left(\sum_j p_{j,t}c_{j,t}\right) - c_{i,t}$$

$$w_{i,t+1} = w_{i,t}(1+\epsilon)^{R_{i,t}}$$

(a) Prove using induction that the total weight of all experts at any step is at most n. (In particular, although individual weights can go up or down, the total weight never goes up).

(b) Express the weight of expert i at time T in the form of the total expected cost of the algorithm and the total cost of the expert. Use part (a) to prove that $\text{cost}_i(\text{ALG}) \leq (1+\epsilon)\text{cost}_i(i) + O(\frac{1}{\epsilon}\log n)$.