COMS 4995 (Randomized Algorithms): Problem Set #5

Due by 11:59 PM on Wednesday, December 4, 2019

Instructions:

(1) Form a group of 1-3 students. You should turn in only one write-up for your entire group.

(2) Submission instructions: We are using Gradescope for the homework submissions. Go to www.gradescope.com to either login or create a new account. Use the course code 9D6V5E to register for this class. Only one group member needs to submit the assignment. When submitting, please remember to add all group member names in Gradescope. See the course Web site for detailed instructions.

(3) Please type your solutions if possible and we encourage you to use the LaTeX template provided on the course home page.

(4) Write convincingly but not excessively.

(5) Some of these problems are difficult, so your group may not solve them all to completion. In this case, you can write up what you've got (subject to (3), above): partial proofs, lemmas, high-level ideas, counterexamples, and so on.

(6) Except where otherwise noted, you may refer to your lecture notes and the specific supplementary readings listed on the course Web page *only*. You can also review any relevant materials from your undergraduate algorithms course. If you do use any approved sources, make you sure you cite them appropriately, and make sure that all your words are your own.

(7) You can discuss the problems verbally at a high level with other groups. And of course, you are encouraged to contact the course staff (via Piazza or office hours) for additional help.

(8) If you discuss solution approaches with anyone outside of your group, you must list their names on the front page of your write-up.

(9) Refer to the course Web page for the late day policy and the School of Engineering honor code.

Problem 25

(16 points) A *zero-sum game* is specified by an $m \times n$ matrix A with real-valued entries in $[0, 1]$. There are two players, a row player and a column player. A choice of a row i and column j specifies an *outcome* of the game, with the entry A_{ij} indicating the payoff of the row player and the negative payoff of the column player in that outcome. For example, Rock-Paper-Scissors can be modeled as a zero-sum game with the following payoff matrix:

	Rock	Paper	Scissors
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

A *strategy* of a player is a probability distribution over rows (for the row player) or columns (for the column player). With row and column strategies $\mathbf{p} \in \mathbb{R}^m$ and $\mathbf{q} \in \mathbb{R}^n$, the expected payoff to the row player is $\mathbf{p}^\top A\mathbf{q}$ (as you should check). A strategy is called *k-sparse* if it has at most k non-zero entries—i.e., randomizes over at most k different rows or columns.

In a zero-sum game, would you rather commit to your strategy before or after the other player commits to theirs? Going last can't possibly hurt you, since you could always play the strategy that you would have played had you gone first (and meanwhile you now have the opportunity to respond to your opponent's strategy, which would seem to be helpful). In other words:

$$\max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{p}^\top A\mathbf{q} \leq \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{p}^\top A\mathbf{q}.$$

(Again, because one option in the inner max on the right-hand side is to always copy the strategy in the outer max of the left-hand side.)

But what if you have to go first?

(a) (12 points) Use the regret guarantee of the FTPL algorithm (Lecture #17) to prove the following: There exists a constant $c > 0$ such that, for every $\epsilon > 0$, there exists a $\frac{c \ln m}{\epsilon^2}$-sparse row strategy $\hat{\mathbf{p}}$ that satisfies

$$\min_{\mathbf{q}} \hat{\mathbf{p}}^\top A\mathbf{q} \geq \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{p}^\top A\mathbf{q} - \epsilon.$$

That is, going first and playing $\hat{\mathbf{p}}$ guarantees the row player an expected payoff almost as large (up to ϵ) as what they would obtain when going last (and playing optimally).

(b) (4 points) Conclude the famous Minimax theorem of von Neumann: for every zero-sum game A,

$$\max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{p}^\top A\mathbf{q} = \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{p}^\top A\mathbf{q}.$$

In other words, as long you and your opponent play optimally, it doesn't matter whether you go first or last in a zero-sum game.

Problem 26

(22 points) The goal of this problem is to investigate a generalization of the Follow-the-Perturbed Leader (FTPL) algorithm discussed in Lecture #17, and then instantiate this generalization to recover another famous no-regret algorithm.

Consider a finite action set $A = \{1, 2, \ldots, n\}$ and let Δ denote the set of probability distributions over A. For convenience, we work with cost vectors (mapping actions to $[0, 1]$) rather than reward vectors. The regret of an action sequence $a^1, \ldots, a^T$ with respect to cost vectors $c^1, \ldots, c^T$ is defined analogously, as the extent to which the cumulative cost incurred along the chosen action sequence exceeds that of the best fixed action in hindsight:

$$\sum_{t=1}^{T} c^t(a^t) - \min_{a \in A} \sum_{t=1}^{T} c^t(a).$$

Let R denote a real-valued function on Δ, called a *regularizer*. This term plays the role of the "fictitious bonuses" in the FTPL algorithm in Lecture #17. For a probability distribution p over A and cost function c^t defined on A, we abuse notation and write $c^t(p)$ for p's expected cost $\mathbf{E}_{a \sim p}[c^t(a)]$. The *follow-the-regularized-leader (FTRL) algorithm* then chooses the distribution over actions that minimizes the cumulative expected cost so far plus the regularizer:

1. At time $t = 1, 2, \ldots, T$:

   (a) Choose p^t in

$$\operatorname*{argmin}_{p \in \Delta} \left(R(p) + \sum_{s=1}^{t-1} c^s(p) \right).$$

(As usual, the adversary chooses the cost vector c^t after observing the algorithm's distribution p^t but before seeing nature's sample a^t from p^t.)

(a) (6 points) Prove that, for every sequence $c^1, \ldots, c^T$ and fixed distribution p^*, the action distribution sequence $p^1, \ldots, p^T$ chosen by FTRL satisfies[1]

$$\sum_{t=1}^{T} c^t(p^t) - \sum_{t=1}^{T} c^t(p^*) \leq \left[\sum_{t=1}^{T} \left(c^t(p^t) - c^t(p^{t+1})\right)\right] + \left[R(p^*) - R(p^1)\right].$$

[Hint: Induction. It might be helpful to start with the special case where R is the all-zero function.]

(b) (3 points) For the rest of the problem, take the regularizer R as the negative entropy function (where $\epsilon > 0$ is a parameter to be chosen later):

$$R(p) := \frac{1}{\epsilon} \sum_{i=1}^{n} p_i \ln p_i,$$

with the convention that $0 \ln 0$ is 0.

With this choice of R, what distribution p^1 will the FTRL algorithm choose in the very first time step? (Prove your answer.)

(c) (2 points) Prove that, for every $p^* \in \Delta$, $R(p^*) - R(p^1) \leq \frac{1}{\epsilon} \ln n$.

(d) (2 points) Prove that $c^t(p^t) - c^t(p^{t+1}) \leq 1 - e^{\epsilon}$ for all t.

[Hint: recall all costs are in $[0, 1]$.]

(e) (4 points) Prove that, with a judicious choice of the parameter ϵ, the FTRL algorithm (with the negative entropy regularizer) has expected regret $O(\sqrt{T \ln n})$, thus matching the (best-possible) bound for FTPL given in Lecture #17.

(f) (5 points) The following algorithm has various names, including "Hedge" and "Exponential Weights."[2]

1. At time 0:
   (a) Initialize $w_a^1 := 1$ for every action $a \in A$.
2. At time $t = 1, 2, \ldots, T$:
   (a) Before seeing the current cost vector c^t: choose the action distribution p^t proportional to the current weight vector w^t (i.e., with each p_a^t equal to $w_a^t / (\sum_{a' \in A} w_{a'}^t)$).
   (b) After seeing the current cost vector c^t: update weights using the formula

$$w_a^{t+1} := w_a^t \cdot e^{-\epsilon c^t(a)} \tag{1}$$

   for every action $a \in A$.

Prove that this algorithm is exactly the same as the FTRL algorithm with the negative entropy regularizer.

Problem 27

(10 points) This goal of this problem is to bound the cover time of a graph in terms of its worst-case hitting time. Precisely, let $G = (V, E)$ be an n-vertex connected undirected graph. As in lecture, let h_{uv} denote the expected number of steps required by a random walk to reach vertex v when started at the vertex u. Recall that the cover time of a graph (with respect to an initial vertex s) is the expected number of steps required

[1] Interpretation (ignoring the extra loss caused by R): the regret of FTRL can only be large if the algorithm is frequently changing its chosen distribution a lot between consecutive time steps. Intuition: a smart choice of regularizer should force the algorithm to change its chosen probability distributions fairly slowly over time.

[2] The "Multiplicative Weights" algorithm is a slightly different algorithm in the same vein, with the update rule in (1) replaced by the similar rule $w_a^{t+1} := w_a^t \cdot (1 - \epsilon c^t(a))$.

by a random walk to visit every vertex at least once when started at s. Prove that the worst-case cover time (over s) is at most an $O(\log n)$ factor larger than the worst-case (over u, v) hitting time:

$$\max_{s \in V} C_s \leq O(\log n) \cdot \max_{u,v \in V \,:\, u \neq v} h_{uv}.$$

[Hint: Order the vertices uniformly at random. Define T_j as the number of steps before visiting the first j vertices. Under what conditions is $T_j - T_{j-1}$ non-zero?]

Problem 28

(15 points) As in Lecture #20, let $\lambda_2(G)$ denote the second-smallest eigenvalue of the Laplacian matrix of an undirected graph G with at least two vertices. This problem investigates properties of λ_2.

(a) (8 points) For a connected graph $G = (V, E)$, let $d(u, v)$, the unweighted distance between vertices $u, v \in V$, be the minimum number of edges in a u-v path. Let $diam(G)$, the diameter of G, be the maximum of $d(u, v)$ over all pairs of vertices u, v.

Prove that if G is connected, then

$$diam(G) \geq \frac{1}{n} \cdot \frac{1}{\lambda_2(G)}.$$

[Hint: use the variational characterization of λ_2.]

(b) (7 points) Assuming G is a connected graph, how small can $\lambda_2(G)$ be as a function of the number of vertices n?

Specifically, let $\bar{\lambda}(n)$ denote the minimum value of $\lambda_2(G)$, over all connected n-vertex graphs G. Find a function $f(\cdot)$ and positive constants $c_1 > c_2 > 0$ such that

$$c_1 f(n) \geq \bar{\lambda}(n) \geq c_2 f(n)$$

for all $n \geq 2$.

Problem 29

(17 points) The point of this problem is to illustrate how (approximately) counting the number of combinatorial objects of a given type often reduces to the problem of sampling a uniformly random such object. Such approximate counting algorithms provide the motivation for much of the work in theoretical computer science on Markov Chain Monte Carlo.

Suppose you are given a black box that takes an input graph G, and returns an independent set of G drawn uniformly at random from among all independent sets of G.[3] Design an algorithm, using this black box, that takes an input graph G, together with parameters $\delta > 0$ and $\epsilon > 0$, and does the following: with probability at least $1 - \delta$ it estimates the number of independent sets in G to within a factor of $1 \pm \epsilon$. The running time of your algorithm should be polynomial in $\frac{1}{\delta}$, $\frac{1}{\epsilon}$, and the size of the input graph; a call to the black box should be considered to take a single computational step.

[Hints: Order the edges arbitrarily and let G_i denote G with only the first i edges present. Let $I(H)$ denote the set of independent sets of a graph H. Rather than estimate $|I(G)|$ directly, estimate (using sampling) each of the ratios $|I(G_i)|/|I(G_{i-1})|$ and take their product. To bound the number of samples needed, determine how small each of these ratios might be.]

[3] Recall that an *independent set* is the complement (in terms of edges) of a clique—a subset S of vertices such that $(u, v) \notin E$ for all $u, v \in E$.

Problem 30

(20 points) There is a small but interesting improvement that can be made to the balanced allocation scheme described in Lecture #21. Again we will place n balls into n bins. We assume for this problem that n is even. Suppose that we divide the n bins into two groups of size $n/2$, the left group and the right group. For each ball, we independently choose one bin uniformly at random from the left group and one bin uniformly at random from the right group. We put the ball in the least loaded bin, but if there is a tie we always put the ball in the bin from the left group. In this problem, you will prove that this scheme reduces the expected maximum load to $(\ln \ln n)/(2 \ln \phi) + O(1)$, where $\phi = (1 + \sqrt{5})/2$ is the golden ratio. This improves the result from lecture by a constant factor.[4]

The key idea in how the proof from lecture must change is that we now require two sequences, β_i and γ_i. Similar to the proof from lecture, β_i represents a desired upper bound on the number of bins on the left with load at least i, and γ_i is a desired upper bound on the number of bins on the right with load at least i. Argue that choosing

$$\beta_{i+1} = \frac{c_1 \beta_i \gamma_i}{n}$$

and

$$\gamma_{i+1} = \frac{c_2 \beta_{i+1} \gamma_i}{n}$$

for some positive constants c_1 and c_2 is sufficient (as long as β_i and γ_i are large enough that Chernoff bounds apply).

Now let F_k denote the kth Fibonacci number. Apply induction to show that, for sufficiently large i, $\beta_i \leq n c_3 c_4^{F_{2i}}$ and $\gamma_i \leq n c_3 c_4^{F_{2i+1}}$ for some constants c_3 and c_4. Following the proof from lecture, use this to prove the $(\ln \ln n)/(2 \ln \phi) + O(1)$ upper bound.

[4] We didn't prove it, but it turns out that the bound of $(\ln \ln n)/(\ln 2) + O(1)$ from Lecture #21 for the original scheme is tight up to the additive constant.