CS261: Problem Set #3*

Due by 11:59 PM on Tue, Feb 25, 2020

Instructions:

(1) We encourage you to pair up with another student in working on this homework. You should turn in only one write-up for your group.

(2) Submission instructions: We will use Gradescope for the homework submissions. Go to www.gradescope.com to either login or create a new account. Use the course code 9GZK2R to register for CS261. Only one group member needs to submit the assignment. When submitting, please remember to add all group member names in Gradescope.

(3) You must use LaTeX, LyX, Microsoft Word, or a similar editor to typeset your write-up.

(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 the materials listed on the course web page *only*. You can also review any relevant materials from your undergraduate algorithms course.

(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.

Problem 13

This problem considers a variant of the online decision-making problem. There are n "experts," where n is a power of 2.

Combining Expert Advice

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

- each expert offers a prediction of the realization of a binary event (e.g., whether a stock will go up or down)
- a decision-maker picks a probability distribution p^t over the possible realizations 0 and 1 of the event
- the actual realization $r^t \in \{0, 1\}$ of the event is revealed
- a 0 or 1 is chosen according to the distribution p^t, and a *mistake* occurs whenever it is different from r^t

*We thank Tim Roughgarden for permission to include problems from his Winter 2016 edition of CS 261.

You are promised that there is at least one omniscient expert who makes a correct prediction at every time step.

(a) Prove that the minimum worst-case number of mistakes that a deterministic algorithm can make is precisely $\log_2 n$ (i.e. give both a lower bound argument and an algorithm that achieves this bound).

(b) Prove that the minimum worst-case expected number of mistakes that a randomized algorithm can make is precisely $\frac{1}{2}\log_2 n$ (again, give both a lower bound and an algorithm).

Problem 14

In lecture we saw that the follow-the-leader (FTL) algorithm, and more generally every deterministic algorithm, can have regret that grows linearly with T. This problem outlines a randomized variant of FTL, the *follow-the-perturbed-leader (FTPL)* algorithm, with worst-case regret comparable to that of the multiplicative weights algorithm. In the description of FTPL, we define each probability distribution p^t over actions implicitly through a randomized subroutine.

Follow-the-Perturbed-Leader (FTPL) Algorithm

for each action $a \in A$ **do**
    independently sample a geometric random variable with parameter η,[1] denoted by X_a
for each time step $t = 1, 2, \ldots, T$ **do**
    choose the action a that maximizes the perturbed cumulative reward $X_a + \sum_{u=1}^{t-1} r^u(a)$ so far

For convenience, assume that, at every time step t, there is no pair of actions whose (unperturbed) cumulative rewards-so-far differ by an integer.

(a) Prove that, at each time step $t = 1, 2, \ldots, T$, with probability at least $1 - \eta$, the largest perturbed cumulative reward of an action prior to t is more than 1 larger than the second-largest such perturbed reward.

[Hint: Sample the X_a's gradually by flipping coins only as needed, pausing once the action a^* with largest perturbed cumulative reward is identified. Resuming, only X_{a^*} is not yet fully determined. What can you say if the next coin flip comes up "tails?"]

(b) As a thought experiment, consider the (unimplementable) algorithm that, at each time step t, picks the action that maximizes the perturbed cumulative reward $X_a + \sum_{u=1}^{t} r^u(a)$ over $a \in A$, *taking into account the current reward vector*. Prove that the regret of this algorithm is at most $\max_{a \in A} X_a$.

[Hint: Consider first the special case where $X_a = 0$ for all a. Iteratively transform the action sequence that always selects the best action in hindsight to the sequence chosen by the proposed algorithm. Work backward from time T, showing that the reward only increases with each step of the transformation.]

(c) Prove that $\mathbf{E}[\max_{a \in A} X_a] \leq b\eta^{-1} \ln n$, where n is the number of actions and $b > 0$ is a constant independent of η and n.

[Hint: use the definition of a geometric random variable and remind yourself about "the union bound."]

(d) Prove that, for a suitable choice of η, the worst-case expected regret of the FTPL algorithm is at most $b\sqrt{T \ln n}$, where $b > 0$ is a constant independent of n and T.

[1] Equivalently, when repeatedly flipping a coin that comes up "heads" with probability η, count the number of flips up to and including the first "heads."

Problem 15

Alice is a student in CS 261 and a budding photographer. She decides to develop her hobby and earn some extra money on the side photographing events and doing photo shoots at Stanford. Of course, she has to upgrade her amateur equipment and needs a good camera and lens to get started. Suppose the cost of buying the equipment is $2000. She also has the option of renting a camera and lens and the cost of this is a much more affordable $200 per assignment. Having started her new photography business, she is now faced with the following RENT OR BUY problem: each time she gets a new assignment, should she rent the camera for the assignment or buy it? Assume that once she buys the equipment, it is good for all future assignments.

What strategy should Alice use for the online problem she faces? Consider two extreme strategies:

1. She could decide to buy the camera when she gets her first assignment, but she is not sure how many assignments she will actually get. If she is unlucky and ends up with only one assignment, then buying the equipment would be a waste of money in retrospect (ignore the resale value in this problem).

2. She could decide to rent always and never buy. If she gets a very large number of assignments, she will run up a large bill with her rental costs. In retrospect, she ought to have bought the camera.

In this problem, you will determine a good strategy for Alice to use. Henceforth, for convenience, assume that the cost of renting is 1 unit and the cost of buying is B units. Let A_i be the deterministic algorithm that rents a maximum of $i-1$ times and buys if there is a ith assignment. (Every deterministic algorithm for this problem is of the form A_i for some $i \geq 1$).

(a) Show that there is a deterministic algorithm with competitive ratio ≤ 2.

(b) Show a lower bound of $2 - 1/B$ on the competitive ratio of any deterministic algorithm.

(c) Consider a randomized algorithm that initially chooses index i at random and then runs algorithm A_i. Say that we pick i with probability p_i. Here $p_i \geq 0$ and $\sum_{i \geq 1} p_i = 1$. (In fact, any randomized algorithm for this problem is of this form). We will determine good values for the p_is later.

Suppose Alice gets a total of j assignments. Write down an expression for the expected cost incurred by the randomized algorithm. Let c denote the competitive ratio of this algorithm. Write down a linear program to determine the best choice of p_i values so as to optimize the competitive ratio of the randomized algorithm. For now, your LP will have an infinite number of variables and constraints, but we will fix this shortly.

(d) Argue that without loss of generality, an optimal randomized strategy has $p_i = 0$ for all $i > B$. Further argue that it suffices to write the LP in the previous part with constraints for the scenarios where Alice gets j assignments, $j <= B$. A brief explanation with a few sentences for each of these arguments is enough to earn full credit for this part. Now we have a bounded size linear program to find the optimal p_i values.

(e) Write down the dual of the (finite) LP from the previous part.

(f) Argue that the dual has the following interpretation: Construct a probability distribution $\mathcal{D}$ on the number of assignments that Alice gets, such that for every deterministic algorithm A_i, the ratio of the expected cost of A_i to the expected value of the optimal cost in hindsight is high. Show that the value of the dual is the minimum of these ratios for algorithms A_i. Your goal is to specify a distribution on the number of assignments from the dual solution, i.e. specify the probability q_j that Alice gets exactly j assignments. The expectations (of the cost of A_i and the optimal cost) referred to earlier are computed w.r.t. this distribution you construct.

This is an instance of **Yao's principle** which is a very useful technique for establishing bounds on the performance of randomized algorithms. This in turn is established via the minimax principle that we have encountered frequently in class.

(g) It turns out the optimum solution to the LP in part (d) satisfies all the constraints with equality. Assuming this, construct an optimal solution to the LP. What are the optimal values for p_i and the best possible competitive ratio?

For this part, it may help to write out the equations explicitly for a small value of B, say $B = 4$. You will notice that consecutive equations have similar coefficients. Consider subtracting one from the other to get a new system of equations.

Problem 16

In this problem we'll show that there is no online algorithm for the online bipartite matching problem with competitive ratio better than $1 - \frac{1}{e} \approx 63.2\%$.

Consider the following probability distribution over online bipartite matching instances. There are n left-hand side vertices L, which are known up front. Let π be an ordering of L, chosen uniformly at random. The n vertices of the right-hand side R arrive one by one, with the ith vertex of R connected to the last $n - i + 1$ vertices of L (according to the random ordering π).

(a) Explain why $OPT = n$ for every such instance.

(b) Consider an arbitrary deterministic online algorithm $\mathbf{A}$. Prove that for every $i \in \{1, 2, \ldots, n\}$, the probability (over the choice of π) that $\mathbf{A}$ matches the ith vertex of L (according to π) is at most

$$\min\left\{\sum_{j=1}^{i} \frac{1}{n-j+1}, 1\right\}.$$

[Hint: for example, in the first iteration, assume that $\mathbf{A}$ matches the first vertex of R to the vertex $v \in L$. Note that $\mathbf{A}$ must make this decision without knowing π. What can you say if v does not happen to be the first vertex of π?]

(c) Prove that for every deterministic online algorithm $\mathbf{A}$, the expected (over π) size of the matching produced by $\mathbf{A}$ is at most

$$\sum_{i=1}^{n} \min\left\{\sum_{j=1}^{i} \frac{1}{n-j+1}, 1\right\}, \tag{1}$$

and prove that (1) approaches $n(1 - \frac{1}{e})$ as $n \to \infty$.

[Hint: for the second part, recall that $\sum_{j=1}^{d} \frac{1}{j} \approx \ln d$ (up to an additive constant less than 1). For what value of i is the inner sum roughly equal to 1?]

(d) Extend (c) to randomized online algorithms $\mathbf{A}$, where the expectation is now over both π and the internal coin flips of $\mathbf{A}$.

[Hint: use the fact that a randomized online algorithm is a probability distribution over deterministic online algorithms (as flipping all of $\mathbf{A}$'s coins in advance yields a deterministic algorithm).]

(e) Prove that for every $\epsilon > 0$ and (possibly randomized) online bipartite matching algorithm $\mathbf{A}$, there exists an input such that the expected (over $\mathbf{A}$'s coin flips) size of $\mathbf{A}$'s output is no more than $1 - \frac{1}{e} + \epsilon$ times that of an optimal solution.

Problem 17

In this problem, you will analyze an algorithm to maintain a linked list of n elements, using competitive analysis. Requests to the algorithm are of the form Access(x), where x is an item in the list. In response to this request, the algorithm walks the list starting from the first position until the element x is encountered. The algorithm is also allowed to modify the list as it proceeds. In particular, in servicing an Access request,

the accessed element may be moved to any earlier position in the list. We think of the input to the algorithm as the request sequence $\sigma = (\sigma_1, \ldots, \sigma_m)$, where σ_k is the element requested by the kth ACCESS request.

In order to analyze this problem under the lens of competitive analysis, we need to be precise about the costs of various operations, so here is a model for the costs:

1. ACCESS(x) where x is currently at position k in the list costs k. (This models the cost of traversing the linked list until x is encountered). Position numbers start from 1.

2. Moving the accessed element to any earlier position is free.

We will analyze a simple strategy MOVE-TO-FRONT (MTF): After ACCESS(x), move x all the way to the front of the list.

It turns out that computing the optimal strategy for LIST UPDATE is NP-complete. Given that we don't know how to compute the optimal strategy efficiently, it is quite remarkable that we can compare the performance of MTF to that of the optimal strategy to show that it is 2-competitive. Henceforth, when we refer to *the optimal* algorithm, we mean some strategy that achieves minimum cost for the request sequence. If there are multiple such strategies, pick one arbitrarily. Note that the optimal strategy is a function of the entire request sequence. Both MTF and the optimal algorithm start with the linked list in the same configuration.

(a) Let $C_{MTF}(\sigma_k)$ denote the cost of MTF on the kth request and let $C_{OPT}(\sigma_k)$ denote the cost of the optimal algorithm on the kth request. $C_{MTF}(\sigma) = \sum_k C_{MTF}(\sigma_k)$ and $C_{OPT}(\sigma) = \sum_k C_{OPT}(\sigma_k)$ One natural approach would be to prove that for every k, $C_{MTF}(\sigma_k) \leq 2C_{OPT}(\sigma_k)$. Show that such a strong statement is not true.

(b) The analysis involves a *potential function* argument that is quite commonly used in online algorithms. We imagine running MTF and OPT side by side on the request sequence. We will define a non-negative function Φ that we call a potential function. Let Φ_k be the value of the potential function after k requests have been serviced. The initial value is $\Phi_0 = 0$. We will show that

$$\underbrace{C_{MTF}(\sigma_k) + \Phi_k - \Phi_{k-1}}_{\text{amortized cost}} \leq 2 \cdot C_{OPT}(\sigma_k) \tag{2}$$

The LHS of the above inequality is referred to as the *amortized cost*. The potential function Φ helps smooth out the costs of the algorithm over various steps so as to facilitate a comparison with the cost of OPT. Show that (2) implies that $C_{MTF}(\sigma) \leq 2 \cdot C_{OPT}(\sigma)$.

(c) Consider the following potential function: Φ is equal to the number of inverted pairs $\{x, y\}$ in the two lists currently maintained by MTF and OPT. $\{x, y\}$ is an inverted pair if x occurs before y in one list, but y appears before x in the other list. Verify that Φ satisfies the conditions $\Phi \geq 0$, $\Phi_0 = 0$, and prove that (2) holds. This establishes that MTF is 2-competitive.

[Hint: Define

$$S = \{y | y \text{ precedes } \sigma_k \text{ in MTF's list and in OPT's list}\}$$

$$T = \{y | y \text{ precedes } \sigma_k \text{ in MTF's list but not in OPT's list}\}$$

Now obtain bounds on $C_{MTF}(\sigma_k)$ and $C_{OPT}(\sigma_k)$ in terms of S and T.]

(d) Let A be any deterministic algorithm for the problem. For any $\epsilon > 0$, show that the competitive ratio of A is at least $\frac{2n}{n+1} - \epsilon$.

[Hint: Construct a long request sequence that makes A incur very high cost. While computing the best offline strategy is NP-complete, a good low cost heuristic is enough to complete the proof.]

Problem 18

You are the sole participant in an auction to buy a painting. The rules of the auction are as follows. The seller of the painting has some fixed target price $p \geq 1$ in their mind. You, as the participant, are allowed to submit bids to buy the painting. If the bid is lower than p, it is rejected, and you may submit another bid. If the bid is at least p, the bid is accepted and you get the painting. However, in order to discourage you from repetitively sending many bids, the price you pay the seller is the sum of all your bids in this auction (the accepted bid and all the previous rejected ones).

(a) Design a deterministic algorithm that buys the painting and pays at most $4p$.

(b) Design a randomized algorithm that buys the painting and pays at most $3p$ in expectation.

[Hint: randomly modify the starting point of your bidding strategy from part (a).]

Note: One goal of this question is to highlight a more general technique (in online algorithms and algorithms in general): For some problem, suppose that we have an algorithm that achieves some competitive ratio if we know the value of OPT. In this story, p is OPT, and if we knew the value of OPT, we could design a 1-competitive algorithm (just bid p). Then we can design an algorithm for the same problem without the knowledge of OPT by "guessing" different values of OPT as you did in part (a).