CS261: Problem Set #3

Due by 11:59 PM on Tuesday, February 23, 2016

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 9B3BEM 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) 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 the course lecture notes *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 fills in some gaps in our proof sketch of strong linear programming duality.

(a) For this part, assume the version of Farkas's Lemma stated in Lecture #9, that given $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$, exactly one of the following statements holds: (i) there is an $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{x} \geq 0$; (ii) there is a $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y}^T\mathbf{A} \geq 0$ and $\mathbf{y}^T\mathbf{b} < 0$.

Deduce from this a second version of Farkas's Lemma, stating that for $\mathbf{A}$ and $\mathbf{b}$ as above, exactly one of the following statements holds: (iii) there is an $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} \leq \mathbf{b}$; (iv) there is a $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} \geq 0$, $\mathbf{y}^T\mathbf{A} = 0$, and $\mathbf{y}^T\mathbf{b} < 0$.

[Hint: note the similarity between (i) and (iv). Also note that if (iv) has a solution, then it has a solution with $\mathbf{y}^T\mathbf{b} = -1$.]

(b) Use the second version of Farkas's Lemma to prove the following version of strong LP duality: if the linear programs

$$\max \mathbf{c}^T\mathbf{x}$$

subject to

$$\mathbf{Ax} \leq \mathbf{b}$$

with $\mathbf{x}$ unrestricted, and

$$\min \mathbf{b}^T\mathbf{y}$$

subject to

$$\mathbf{A}^T\mathbf{y} = \mathbf{c}, \mathbf{y} \geq 0$$

are both feasible, then they have equal optimal objective function values.

[Hint: weak duality is easy to prove directly. For strong duality, let γ^* denote the optimal objective function value of the dual linear program. Add the constraint $\mathbf{c}^T\mathbf{x} \geq \gamma^*$ to the primal linear program and use Farkas's Lemma to show that the feasible region is non-empty.]

Problem 14

Recall the *multicommodity flow* problem from Exercise 17. Recall the input consists of a directed graph $G = (V, E)$, k "commodities" or source-sink pairs $(s_1, t_1), \ldots, (s_k, t_k)$, and a positive capacity u_e for each edge.

Consider also the *multicut* problem, where the input is the same as in the multicommodity flow problem, and feasible solutions are subsets $F \subseteq E$ of edges such that, for every commodity (s_i, t_i), there is no s_i-t_i path in $G = (V, E \setminus F)$. (Assume that s_i and t_i are distinct for each i.) The *value* of a multicut F is just the total capacity $\sum_{e \in F} u_e$.

(a) Formulate the multicommodity flow problem as a linear program with one decision variable for each path P that travels from a source s_i to the corresponding sink t_i. Aside from nonnegativity constraints, there should be only be m constraints (one per edge).

[Note: this is a different linear programming formulation than the one asked for in Exercise 21.]

(b) Take the dual of the linear program in (a). Prove that every optimal 0-1 solution of this dual — i.e., among all feasible solutions that assign each decision variable the value 0 or 1, one of minimum objective function value — is the characteristic vector of a minimum-value multicut.

(c) Show by example that the optimal solution to this dual linear program can have objective function value strictly smaller than that of every 0-1 feasible solution. In light of your example, explain a sense in which there is no max-flow/min-cut theorem for multicommodity flows and multicuts.

Problem 15

This problem gives a linear-time (!) randomized algorithm for solving linear programs that have a large number m of constraints and a small number n of decision variables. (The constant in the linear-time guarantee $O(m)$ will depend exponentially on n.)

Consider a linear program of the form

$$\max \mathbf{c}^T\mathbf{x}$$

subject to

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}.$$

For simplicity, assume that the linear program is feasible with a bounded feasible region, and let M be large enough that $|x_j| < M$ for every coordinate of every feasible solution. Assume also that the linear program is "non-degenerate," in the sense that no feasible point satisfies more than n constraints with equality. For example, in the plane (two decision variables), this just means that there does not exist three different constraints (i.e., halfplanes) whose boundaries meet at a common point. Finally, assume that the linear program has a unique optimal solution.[1]

Let $C = \{1, 2, \ldots, m\}$ denote the set of constraints of the linear program. Let B denote additional constraints asserting that $-M \leq x_j \leq M$ for every j. The high-level idea of the algorithm is: (i) drop a

[1] All of these simplifying assumptions can be removed without affecting the asymptotic running time; we leave the details to the interested reader.

random constraint and recursively compute the optimal solution $\mathbf{x}^*$ of the smaller linear program; (ii) if $\mathbf{x}^*$ is feasible for the original linear program, return it; (iii) else, if $\mathbf{x}^*$ violates the constraint $\mathbf{a}_i^T\mathbf{x} \le b_i$, then change this inequality to an equality and recursively solve the resulting linear program.

More precisely, consider the following recursive algorithm with two arguments. The first argument C_1 is a subset of inequality constraints that must be satisfied (initially, equal to C). The second argument is a subset C_2 of constraints that must be satisfied with equality (initially, $\emptyset$). The responsibility of a recursive call is to return a point maximizing $\mathbf{c}^T\mathbf{x}$ over all points that satisfy all the constraints of $C_1 \cup B$ (as inequalities) and also those of C_2 (as equations).

```
Linear-Time Linear Programming

Input:  two disjoint subsets $C_1, C_2 \subseteq C$ of constraints
Base case #1:  if $|C_2| = n$, return the unique point that satisfies every constraint
  of $C_2$ with equality
Base case #2:  if $|C_1| + |C_2| = n$, return the point that maximizes $\mathbf{c}^T\mathbf{x}$ subject to
  $\mathbf{a}_i^T\mathbf{x} \le b_i$ for every $i \in C_1$, $\mathbf{a}_i^T\mathbf{x} = b_i$ for every $i \in C_2$, and the constraints in $B$
Recursive step:
choose $i \in C_1$ uniformly at random
recurse with the sets $C_1 \setminus \{i\}$ and $C_2$ to obtain a point $\mathbf{x}^*$
if $\mathbf{a}_i^T\mathbf{x}^* \le b_i$ then
    return $\mathbf{x}^*$
else
    recurse with the sets $C_1 \setminus \{i\}$ and $C_2 \cup \{i\}$, and return the result
```

(a) Prove that this algorithm terminates with the optimal solution $\mathbf{x}^*$ of the original linear program.

[Hint: be sure to explain why, in the "else" case, it's OK to recurse with the ith constraint set to an equation.]

(b) Let $T(m, s)$ denote the expected number of recursive calls made by the algorithm to solve an instance with $|C_1| = m$ and $|C_2| = s$ (with the number n of variables fixed). Prove that T satisfies the following recurrence:

$$T(m,s) = \begin{cases} 1 & \text{if } s = n \text{ or } m + s = n \\ T(m-1,s) + \frac{n-s}{m} \cdot T(m-1,s+1) & \text{otherwise.} \end{cases}$$

[Hint: you should use the non-degeneracy assumption in this part.]

(c) Prove that $T(m, 0) \le n! \cdot m$.

[Hint: it might be easiest to make the variable substitution $\delta = n - s$ and proceed by simultaneous induction on m and δ.]

(d) Conclude that, for every fixed constant n, the algorithm above can be implemented so that the expected running time is $O(m)$ (where the hidden constant can depend arbitrarily on n).

Problem 16

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 $r^t$
```

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 $\log_2 n$.

(b) Prove that the minimum worst-case expected number of mistakes that a randomized algorithm can make is precisely $\frac{1}{2} \log_2 n$.

Problem 17

In Lecture #11 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 $\eta$,[2] 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?"]

[2] 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."

(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.

Problem 18

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.