Homework 2

Out: *Sep 27* Due: *Oct 10*

Instructions:

- Upload your solutions (to the non-extra-credit) to each problem as a **single PDF** file (one PDF total) to Gradescope. Please make sure you are uploading the correct PDF! Please anonymize your submission (i.e., do not list your name in the PDF), but if you forget, it's OK.
- If you choose to do extra credit, upload your solution to the extra credits as a single separate PDF file to Gradescope. Please again anonymize your submission.
- You may collaborate with any classmates, textbooks, the Internet, etc. Please upload a brief "collaboration statement" listing any collaborators as a separate PDF on Gradescope (if you forget, it's OK). But always **write up your solutions individually**.
- For each problem, you should have a solid writeup that clearly states key, concrete lemmas towards your full solution (and then you should prove those lemmas). A reader should be able to read any definitions, plus your lemma statements, and quickly conclude from these that your outline is correct. This is the most important part of your writeup, and the precise statements of your lemmas should tie together in a correct logical chain.
- A reader should also be able to verify the proof of each lemma statement in your outline, although it is OK to skip proofs that are clear without justification (and it is OK to skip tedious calculations). Expect to learn throughout the semester what typically counts as 'clear'.
- You can use the style of Lecture Notes and Staff Solutions as a guide. These tend to break down proofs into roughly the same style of concrete lemmas you are expected to do on homeworks. However, they also tend to prove each lemma in slightly more detail than is necessary on PSets (for example, they give proofs of some small claims/observations that would be OK to state without proof on a PSet).
- Each problem is worth twenty points (even those with multiple subparts), unless explicitly stated otherwise.

Problems:

§1 Given black-box access to a poly-time algorithm $\mathcal{A}_P$ that optimizes linear functions over the convex, compact region P, and poly-time $\mathcal{A}_Q$ that optimizes linear functions over the convex, compact region Q, design a poly-time algorithm that optimizes linear functions over the convex, compact region $P \cap Q$.

§2 The maximum cut problem asks us to cluster the nodes of a graph $G = (V, E)$ into two disjoint sets X, Y so as to maximize the number of edges between these sets:

$$\max_{X,Y} \sum_{(i,j)\in E} \mathbb{1}[(i \in X, j \in Y) \vee (i \in Y, j \in X)]$$

Consider instead clustering the nodes into **three** disjoint sets X, Y, Z. Our goal is to maximize the number of edges between different sets:

$$\max_{X,Y,Z} \sum_{(i,j)\in E} \mathbb{1}[(i \in X, j \in Y \cup Z) \vee (i \in Y, j \in X \cup Z) \vee (i \in Z, j \in X \cup Y)]$$

Design an algorithm based on SDP relaxation that solves this problem with approximation ration greater then .7.

Note: In the Goemans-Williamson algorithm for maximum cut, we claimed that $\frac{2\theta}{\pi(1-\cos\theta)} \geq 0.878$, $\forall\theta \in [0, \pi]$. This is much easier to verify analytically (e.g. with a plot in MATLAB) than to prove formally. If similar quantities appear in your proof, feel free to bound them analytically, without proof.

Obtain the highest object value you can – partial credit will be given to any non-trivial solution, even if it obtains a weaker bound than .7.

§3 (Discrepancy Theory) Suppose you are given a matrix $A \in \{0, 1\}^{n\times n}$ such that each column of A has at most s ones. Our goal is to *color* each column with -1 or $+1$ so that when we take a linear combination of the columns (multiplied by their color), we get a vector where each coordinate is $O(s)$. That is, we are hoping to color each column in a way so that all rows have a nearly-equal number of zeroes and ones.

Formally, we seek a vector (coloring) $\vec{\varepsilon} \in \{-1, +1\}^n$ such that $\|A\vec{\varepsilon}\|_\infty = O(s)$. Observe that if $\vec{a}_t$ is the t-th column of A, we can equivalently write: $\|A\vec{\varepsilon}\|_\infty := \|\sum_{t=1}^n \vec{a}_t \varepsilon_t\|_\infty := \max_{i=1}^n\{|a_{ti}\varepsilon_t|\} = \max_{i=1}^n\{|A_{it}\varepsilon_t|\}$.

We will analyze the following algorithm to achieve this task. The algorithm will iteratively color more and more columns of A. Initially, all columns are "uncolored" and a column $\vec{a}_t$ gets "colored" when ε_t is defined.

- In any iteration, if the number of remaining uncolored columns is at most $2s$, color them arbitrarily and halt.

- Otherwise, we solve the following feasibility LP.

$$\begin{aligned}
\text{Variables: } & x_t \ \forall\ t \in [n]. \\
\text{Constraints: } & x_t \in [-1,1] \ \forall \text{ uncolored } t. \\
& x_t = \varepsilon_t \ \forall \text{ previously colored } t. \\
\sum_t A_{it} x_t &= 0 \ \forall\ i \text{ such that row } i \text{ has} \geq 2s \text{ uncolored ones.}
\end{aligned}$$

  That is, we are allowed to fractionally color each column in $[-1,1]$. However, we must respect any previously colored columns. Moreover, if any row has many uncolored ones, we must ensure that the (fractional) coloring results in a 0 for that row.

- Find a *basic feasible solution* x^* of the above LP, and for any uncolored column t that gets $x_t^* \in \{-1,+1\}$ in this basic solution, we permanently set its ε_t to that color. A *basic feasible solution* of an LP is one that is an extreme point of the feasible space. That is, it cannot be written as a convex combination of other feasible points. You may use without proof that any basic feasible solution of the above LP makes exactly n constraints tight (hold with equality).[1]
- Repeat.

(a) Prove that in every iteration, the LP has a feasible solution.

(b) Prove that in every iteration, the number of row constraints in the LP is at most half the number of currently uncolored columns.

Conclude that a basic feasible solution will integrally color at least half of the uncolored columns in each iteration.

(c) Prove that this algorithm obtains a solution with discrepancy $O(s)$.

Remark: There is nothing special about $A \in \{0,1\}^{n\times n}$ vs. $A \in \{0,1\}^{n\times T}$, i.e., having $T \gg n$ columns. We can easily extend the above result to T columns using the idea of finding a basic solution in the beginning with at most n uncolored columns.

§4 Let $X = \{x_1, \ldots, x_n\}$ be n (not necessarily distinct) unknown numbers in $[0,1]$. You may repeatedly *sample with replacement* a uniformly random element of X.

(a) Prove that no algorithm using $o(n)$ samples can estimate the value of the median within a factor of 1.1. That is, any algorithm which (possibly randomly) maps $m = o(n)$ samples to a guess at the median is off by a factor of at least 1.1 with probability at least 1/3.

Hint: come up with two instances with very different medians, but which look the same after $o(n)$ samples with high probability. To prove that your algorithm must behave similarly on these instances, recall that your algorithm's behavior is completely determined by the samples it sees (you can avoid tedious calculations by cleverly using this observation).

[1] Such a solution can be found by picking a uniformly random vector $\vec{c}$, and optimizing $\vec{c} \cdot \vec{x}$ over the feasible region.

(b) Instead, say we seek a number y such that at least $n/2-t$ elements of X exceed y, and $n/2-t$ numbers are less than y. Prove that if we take $m = O((n/t)^2 \ln(1/\delta))$ samples, and let y denote the median of the m samples, then y has this property with probability at least $1-\delta$.

§5 (a) Consider the following random process: there are n coupons $\{1, \ldots, n\}$. Each step, you draw a uniformly random coupon independently with replacement, and you repeat this until you have drawn *all coupons in* $\{1, \ldots, n\}$. Prove that with probability at least $1-1/n$, the process takes $O(n \log n)$ steps.

Hint: You may find it easier to use Chernoff bounds and union bounds than previous analysis you've seen related to the coupon collector problem.

(b) Consider the following process for matching n jobs to n processors. In each step, every job picks a processor at random. The jobs that have no contention on the processors they picked get executed, and all the other jobs *back off* and then try again. Jobs only take one round of time to execute, so in every round all the processors are available. Show that all the jobs finish executing w.h.p. after $O(\log \log n)$ steps.

Hint: Try to argue that if there are currently ϵn unmatched jobs, then in the next round roughly $\epsilon^2 n$ jobs remain unmatched.

Extra Credit:

§1 (extra credit) Consider the following problem: there are $n > k$ independent (but not identically distributed) non-negative random variables $X_1, \ldots, X_n$ drawn according to distributions $D_1, \ldots, D_n$. Initially, you know each D_i but none of the X_is.

Starting from $i = 1$, each X_i is revealed one at a time. Immediately after it is revealed, you must decide whether to "accept i" or "reject i," before seeing the next X_{i+1}. You may accept at most k elements in total (that is, once you've accepted k times, you must reject everything that comes after). Your reward at the end is $\sum_{i|i \text{ was accepted}} X_i$.

(a) For general k, design a policy that guarantees expected reward at least $(1 - O(\sqrt{\frac{\ln(k)}{k}})) \cdot \mathbb{E}_{X_1,\ldots,X_n \leftarrow D_1,\ldots,D_n}[\sum_{j=1}^{k} X_{r(j)}]$, where r is a permutation from $[n]$ to $[n]$ satisfying $X_{r(1)} \geq X_{r(2)} \geq \ldots \geq X_{r(n)}$ (i.e. the policy gets expected reward at least $(1 - O(\sqrt{\frac{\ln(k)}{k}}))$ times the expected sum of top k weights, which is the best you could do even if you knew all the weights up front).

Hint: Try to set up a simple policy that can be analyzed using a Chernoff bound.

(b) Come up with an example showing that it is not possible to improve the above guarantee beyond $(1 - \Omega(1/\sqrt{k}))$ (which is optimal - no need to prove this).

Hint: an example exists whose complete proof should fit in half a page. You may use without proof the fact that if X is the number of coin flips which land heads from k independent fair coin flips, then $\mathbb{E}[|X - k/2|] = \Theta(\sqrt{k})$.