PRINCETON UNIVERSITY FALL '18 COS 521:ADVANCED ALGORITHMS

Homework 2 **(60 Points)**

Out: *Oct 11* Due: *Oct 26*

Collaboration is allowed on this problem set, but solutions must be written-up individually. Please list collaborators for each problem separately, or write "No Collaborators" if you worked alone. Collaboration is not allowed on bonus problems.

Please prepare your problem sets in LaTeX and compile to a PDF for your final submission. A LaTeX template is available on the course webpage.

§1 **(12 points)** (Approximate LP Solving via Multiplicative Weights) This exercise develops an algorithm to approximately solve Linear Programs.

Consider the problem of finding if a system of linear inequalities as below admits a solution - i.e., whether the system is feasible. This is an example of a feasibility linear program and while it appears restrictive, one can use it solve arbitrary linear programs to obtain approximate solutions.

$$\begin{aligned} a_1^\top x &\geq b_1 \\ a_2^\top x &\geq b_2 \\ &\vdots \\ a_m^\top x &\geq b_m \\ x_i &\geq 0 \ \forall \ i \in [n] \\ \sum_{i=1}^{n} x_i &= 1. \end{aligned} \tag{1}$$

1) Show a simple method to solve the following linear program with two non-trivial constraints for any weights $w_1, w_2, \ldots, w_m$ (the weights are fixed, and x_is are the variables, as above).

$$\begin{aligned} \max \sum_{j=1}^{m} w_j (a_j^\top x - b_j) & \\ x_i &\geq 0 \ \forall i \in [n] \\ \sum_{i=1}^{n} x_i &= 1. \end{aligned} \tag{2}$$

Conclude that if there are non-negative weights $w_1, w_2, \ldots, w_m$ such that the value of the program above is negative, then the system (1) is infeasible.

2) The above setting of finding weights that certify infeasibility of (1) might remind you of the setting of weighting the experts via multiplicative weights update rule

discussed in the class. Use these ideas to obtain an algorithm that a) either finds a set of non-negative weights certifying infeasibility of LP in (1) or b) finds a solution x that approximately satisfies all the constraints in (1), i.e., for each $1 \leq i \leq m$, $a_j^\top x - b_j \geq -\epsilon$, $x_i \geq 0$ and $\sum_{i=1}^n x_i = 1$. Give a bound, as tight as possible, for the number of update steps required in order to reach the above goal and use it to obtain a running time bound for approximate LP solving. You may use the meta-theorem MW from the lecture as a blackbox.

(Hint: Identify m "experts" - one for each inequality constraint in (1) and maintain a weighting of experts (starting with the uniform weighting of all 1s, say) for times $t = 0, 1, \ldots,$ - these are your progressively improving guesses for the weights. Solve (2) using the weights at time t. If the value of (2) is negative, you are done, otherwise think of the "cost" of the j^{th} expert as $a_j^\top x^{(t)} - b_j$ where $x^{(t)}$ is the solution to the LP (2) at time t and update the weights.)

§2 **(10 points)** In ℓ_2 regression you are given datapoints $x_1, x_2, \ldots, x_n \in \Re^k$ and some values $y_1, y_2, \ldots, y_n \in \Re$ and wish to find the "best"linear function that fits this dataset. A frequent choice for best fit is the one with *least squared error*, i.e. find $a \in \Re^k$ that minimizes

$$\sum_{i=1}^{n} |y_i - a \cdot x_i|^2.$$

Show how to solve this problem in polynomial time (hint: reduce to solving linear equations; at some point you may need a certain matrix to be invertible, which you can assume.).

§3 Recall the max-flow problem from undergraduate algorithm: for a directed graph $G(V, E)$ with non-negative capacities c_e for every $e \in E$ and two special vertices s (source) and t (sink), a *flow* in G is an assignment $f : E \to f_e$ such that $f_e \leq c_e$ for every edge and for every vertex $v \in V$, $\sum_{(u,v)\in E} f((u,v)) = \sum_{(v,u)\in E} f((v,u))$. The task is to find a maximum flow f i.e., a flow f such that $\sum_{(s,u)\in E} f((s,u))$ is maximized.

(a) **(8 points)** Show that the following LP is a valid formulation for computing the maximum flow in G. There is a variable $f(u,v)$ for all $(u,v) \in E$.

$$\begin{aligned} &\max \sum_u f(u,t) \\ &\forall e = (u,v) \in E,\ f(u,v) \leq c_e \\ &\forall v \notin \{s,t\},\ \sum_u f(u,v) = \sum_w f(v,w) \\ &\forall e \in E,\ f(e) \geq 0 \end{aligned} \tag{3}$$

(b) **(8 points)** Write the dual for the LP (3). Show that this dual LP computes the minimum *fractional s-t* cut in G (a cut that separates s and t in G and minimizes the sum of the capacities c_e of the edges going across it. You will know what a

fractional s-t cut is once you take the dual: every node isn't entirely on the s side or the t side, but rather partially on each) Use strong duality (Lecture 7) to conclude the *fractional* max-flow min-cut theorem. That is, if the max-flow is C, there exists a fractional s-t cut of value C, and no fractional s-t cut of value $< C$.

(c) (Extra Credit) Devise a rounding scheme that takes as input a fractional min-cut of value C and outputs a true (deterministic) min-cut of value C.

§4 **(12 points)** In class we designed a 3/4-approximation for MAX-2SAT using LP rounding. The MAX-SAT problem is similar except for the fact that the clauses can contain any number of literals. Formally, the input consists of n boolean variables $x_1, x_2, \ldots, x_n$ (each may be either 0 (false) or 1 (true)), m clauses $C_1, C_2, \ldots, C_m$ (each of which consists of disjunction (an or) of some number variables or their negations) and a non-negative weight w_i for each clause. The objective is to find an assignment of 1 or 0 to x_is that maximize the total weight of satisfied clauses. As we saw in the class, a clause is satisfied if one of its non-negated variable is set to 1, or one of the negated variable is set to 0. You can assume that no literal is repeated in a clause and at most one of x_i or $\neg x_i$ appears in any clause.

(a) Generalize the LP relaxation for MAX-2SAT seen in the class to obtain a LP relaxation of the MAX-SAT problem.

(b) Use the standard randomized rounding algorithm on the LP-relaxation you designed in part (1)) to give a $(1 - 1/e)$ approximation algorithm for MAX-SAT. Note that clauses can be of any length.

(c) A naive algorithm for MAX-SAT problem is to set each variable to true with probability 1/2 (without writing any LP). It is easy to see that this *unbiased randomized* algorithm of MAX-SAT achieves 1/2- approximation in expectation. Show the algorithm that returns the best of two solutions given by the randomized rounding of the LP and the simple unbiased randomized algorithm is a 3/4-approximation algorithm of MAX-SAT.

(d) (Extra Credit) Can you give a different randomized rounding of the LP in part (1) above that achieves 3/4-approximation without using the unbiased rounding?

§5 **(10 points)** (Firehouse location) Suppose we model a city as an m-point finite metric space with $d(x, y)$ denoting the distance between points x, y. These $\binom{m}{2}$ distances (which satisfy triangle inequality) are given as part of the input. The city has n houses located at points $v_1, v_2, \ldots, v_n$ in this metric space. The city wishes to build k firehouses and asks you to help find the best locations $c_1, c_2, \ldots, c_k$ for them, which can be located at any of the m points in the city. The *happiness* of a town resident with the final locations depends upon his distance from the closest firehouse. So you decide to minimize the cost function $\sum_{i=1}^{n} d(v_i, u_i)$ where $u_i \in \{c_1, c_2, \ldots, c_k\}$ is the firehouse closest to v_i. Describe an LP-based algorithm that runs in $poly(m)$ time and solves this problem approximately. If OPT is the optimum cost of a solution with k firehouses, your solution is allowed to use $O(k \log m)$ firehouses and have cost at most $(1 + \epsilon)$OPT.

§6 (extra credit) Design an algorithm that uses k firehouses but has cost $O(\mathrm{OPT})$. (Needs a complicated dependent rounding; you can also try other ideas.) Partial credit available for partial progress.