Collaborators

Ben Nelson, Corbin Baldwin and I collaborated for this assignment.

Question 1: Basics of linear programming ..... **[??]**

Consider the feasible set of the following set of inequalities in two variables x, y:

$$\begin{aligned} x + y &\geq 1, \\ x + 2y &\leq 4, \\ y &\leq 2. \end{aligned}$$

(a) **[3]** Draw the feasible region on the plane (you may also draw on paper, roughly to scale, and attach a scanned image).

(b) **[2]** What is the maximum value of $x + 4y$ subject to (x, y) being in the feasible region?

(c) **[2]** Answer the same for $x + y$.

(d) **[2]** Find a point (x, y) that is the intersection of two of the lines defining the equations above, but is not a corner point of the feasible region.

Question 2: Paths and spanning trees as linear optimization ..... **[??]**

In class, we saw the following idea to write the shortest path problem as linear optimization over a $\{0, 1\}$ (discrete) domain. Suppose we have an undirected graph $G = (V, E)$, and suppose that w_e denotes the weight (or length of edge e). The goal is to find the shortest total length path between s and t. The idea was to choose binary variables x_{ij} that now correspond to *directed edges* (going from $i \to j$), and the idea is that we would set $x_{ij} = 1$ if the shortest path from s to t uses the directed edge (i, j).

(a) **[4]** Formally write down the objective and the set of constraints (discussed roughly in class and in the hint below). Note that you need to argue that any feasible solution to the constraints you write down yields a valid path in the graph.

[*Hint:* every vertex other than s, t must have precisely one edge coming in and going out.]

(b) **[7]** Now consider the minimum spanning tree problem, in which we have an undirected graph with weights on the edges, and the goal is to find the connected subgraph (i.e., one with a path between every pair of vertices in V) that minimizes the total weight of the selected edges.

Question 3: Checking feasibility vs optimization ..... **[??]**

Some of the algorithms for linear programming (e.g. simplex) start off with one of the corner points of the feasible set. This turns out to be tricky in general. In this problem, we will see that in general, finding one feasible point is as difficult as actually performing the optimization!

Consider the following linear program (in n variables $x_1, \ldots, x_n$, represented by the vector x):

$$\begin{aligned} &\text{minimize } c^T x \text{ subject to} \\ &a_1^T x \geq b_1 \\ &a_2^T x \geq b_2 \\ &\cdots \\ &a_m^T x \geq b_m. \end{aligned}$$

(a) **[6]** Suppose you know that the optimum value (i.e. the minimum of $c^T x$ over the feasible set) lies in the interval $[-M, M]$ for some real number M (this is typically possible in practice). Suppose also that you have an **oracle** that can take any linear program and say whether it is feasible or not. Prove that using $O(\log(M/\epsilon))$ calls to the oracle, one can determine the optimum value of the LP above up to an error of $\pm\epsilon$, for any given accuracy $\epsilon > 0$. [*Hint:* can you write a new LP that is feasible only if the LP above has optimum value $\leq z$, for some z?]

(b) [**6**] Part (a) gave a way to find the optimum *value*. Now suppose we wish to find the optimum solution (i.e., the best x). Suppose we knew that in the optimum solution, $x_i \in [-M, M]$ for all i. Show how to find each x_i to an error $\pm\epsilon$ using $O(n \log(M/\epsilon))$ calls to the oracle.

Question 4: Modified regression as a linear program ..... [**??**]

Regression is one of the common problems in data applications. We are given n vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ (in say d-dimensions), and real numbers $b_1, b_2, \ldots, b_n$. The goal is to find an $x \in \mathbb{R}^d$ such that $\langle \mathbf{a}_i, x \rangle \approx b_i$ for all i. In the usual least squares formulation, one tries to find an x so as to minimize $\sum_{i=1}^{n} |\langle \mathbf{a}_i, x \rangle - b_i|^2$.

Suppose we change the objective a little, and we wish to find an x so as to minimize

$$\sum_{i=1}^{n} |\langle \mathbf{a}_i, x \rangle - b_i|.$$

(I.e., the objective from earlier, but without the square). Prove that this optimization can be cast as a linear program of size polynomial in n, d. (You will receive partial credit if you come up with an exponential sized formulation.)

Question 5: Starbucks locations ..... [**??**]

Consider the following "Starbucks location" problem. We are given the locations of m strip malls, that are possible locations for opening a store. We are also given the addresses of n customers. Let d_{ij} be the distance between the i'th customer's address and the location of the jth strip mall. Let us assume that we are given as input this matrix d_{ij}.

Given a radius R, the goal is to find the smallest number of stores that need to be opened, so as to guarantee that every customer is at distance $\leq R$ to at least one of the opened stores.

Let us consider an optimization formulation that uses binary variables $\{y_j\}_{j=1}^{m}$, where y_j indicates if a store is to be opened at location j.

(a) [**3**] Write down the constraints one needs to place on the y_j's, and the objective function to optimize.

(b) [**5**] Suppose we "relax" the constraints $y_j \in \{0, 1\}$ to $0 \leq y_j \leq 1$, thus obtaining a linear program (that we can solve efficiently). Give an example (by drawing a picture of locations and addresses in 2D) where the obtained linear program has a strictly smaller optimum value than that of the original problem.

Design an optimization formulation for this problem, possibly using binary variables. Your formulation should have size polynomial in n, m.

(c) [**4**] Suppose we solve the linear program and find that in the optimal solution, all the variables y_j are either 0 or are ≥ 0.5. Use this y to come up with a feasible way of opening stores such that (a) we satisfy the constraint that every user has an open store at distance $\leq R$, and (b) the number of stores opened is at most twice the optimum number.

References

[1] "Freivalds' Algorithm." *Wikipedia*, Wikimedia Foundation, 30 Sept. 2018, `https://en.wikipedia.org/wiki/Freivalds\%27\_algorithm`.