CS 170 HW 9

Due on 2019-04-01, at 11:59 pm

1 Study Group

List the names and SIDs of the members in your study group.

2 Modeling: Tricks of the Trade

One of the most important problems in the field of *statistics* is the *linear regression problem*. Roughly speaking, this problem involves fitting a straight line to statistical data represented by points – $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ – on a graph. Denoting the line by $y = a + bx$, the objective is to choose the constants a and b to provide the "best" fit according to some criterion. The criterion usually used is the *method of least squares*, but there are other interesting criteria where linear programming can be used to solve for the optimal values of a and b. For each of the following criteria, formulate the linear programming model for this problem:

1. Minimize the sum of the absolute deviations of the data from the line; that is,

$$\min \sum_{i=1}^{n} |y_i - (a + bx_i)|$$

2. Minimize the maximum absolute deviation of the data from the line; that is,

$$\min \max_{i=1\ldots n} |y_i - (a + bx_i)|$$

Solution:

(i) Given n data points (x_i, y_i) for $i = 1, 2, \ldots n$, as well as variables a and b, define new variables $z_i = y_i - (a + bx_i)$ to denote the deviation of the i^{th} data point from the line $y = a + bx$. We know that any number, positive or negative, can be represented by the difference of two positive numbers, so that $z_i = z_i^+ - z_i^-$ and $z_i^+ \geq 0$ and $z_i^- \geq 0$ (so we are also introducing new variables z_i^+ and z_i^-).

Observe that minimizing $|z_i|$ (which is non-linear) is equivalent to minimizing $z_i^+ + z_i^-$ (which is linear) subject to the above three constraints.

Why is this the case? Let's try a small example. We could represent the number 4 as 4 - 0, 5 - 1, 6 - 2, 100 - 96, etc. But notice that the pair that gives us the smallest sum is 4 and 0. And in general for z_i, the z_i^+, z_i^- pair that gives us the smallest sum will always be either $z_i^+ = z_i$ and $z_i^- = 0$ for positive z_i, or $z_i^+ = 0$ and $z_i^- = z_i$ for negative z_i.

Hence

$$\min \sum_{i=1}^{n} |y_i - (a + bx_i)|$$

is equivalent to

$$\text{Minimize } \sum_{i=1}^{n} z_i^+ + z_i^-$$
$$\text{subject to } \begin{cases} y_i - (a + bx_i) = z_i^+ - z_i^- & \text{for } 1 \le i \le n \\ z_i^+ \ge 0 & \text{for } 1 \le i \le n \\ z_i^- \ge 0 & \text{for } 1 \le i \le n \end{cases}.$$

(ii) By the same reasoning, we introduce variables $z_i^+ - z_i^- = y_i - (a + bx_i)$, for $z_i^+ \ge 0$ and $z_i^- \ge 0$. Observe that minimizing $|z_i|$ (which is non-linear) is equivalent to minimizing $\max\{z_i^+, z_i^-\}$ (still non-linear) subject to the above constraints, which is equivalent to minimizing t such that $t \ge z_i^+$ and $t \ge z_i^-$ (which is linear). In simpler words, the smallest possible upper bound on z_i^+ and z_i^- will always be precisely equal to the larger of the two. Hence

$$\min \max_{i=1\dots n} |y_i - (a + bx_i)|$$

is equivalent to

$$\text{Minimize } t$$
$$\text{subject to } \begin{cases} y_i - (a + bx_i) = z_i^+ - z_i^- & \text{for } 1 \le i \le n \\ z_i^+ \ge 0 & \text{for } 1 \le i \le n \\ z_i^- \ge 0 & \text{for } 1 \le i \le n \\ z_i^+ \le t & \text{for } 1 \le i \le n \\ z_i^- \le t & \text{for } 1 \le i \le n \end{cases}.$$

Remark: The following is an alternative solution to (i):

$$\text{Minimize } \sum_{i=1}^{n} t_i$$
$$\text{subject to } \begin{cases} y_i - (a + bx_i) \le t_i & \text{for } 1 \le i \le n \\ y_i - (a + bx_i) \ge -t_i & \text{for } 1 \le i \le n \end{cases}.$$

The following is an alternative solution to (ii):

$$\text{Minimize } t$$
$$\text{subject to } \begin{cases} y_i - (a + bx_i) \le t & \text{for } 1 \le i \le n \\ y_i - (a + bx_i) \ge -t & \text{for } 1 \le i \le n \end{cases}.$$

3 Zero Sum Games

Alice and Bob are playing a zero-sum game whose payoff matrix is shown below. The ij^{th} entry of the matrix shows the payoff that Alice receieves if she plays strategy i and Bob plays strategy j. Alice is the row player and is trying to maximize her payoff.

Alice \Bob	A	B
1	4	1
2	2	5

Now we will write a linear program to find a strategy that maximizes Alice's payoff

(a) The variables of the linear program are x_1, x_2 and p where x_i and p denote ?

(b) Write the linear program for maximizng Alice's payoff.

(c) Eliminate x_2 from the linear program and write it in terms of p and x_1 alone.

(d) Draw the feasible region of the above linear program in p and x_1. You are encouraged to use a plotting software for this.

(e) What is the optimal solution and what is the value of the game?

Solution:

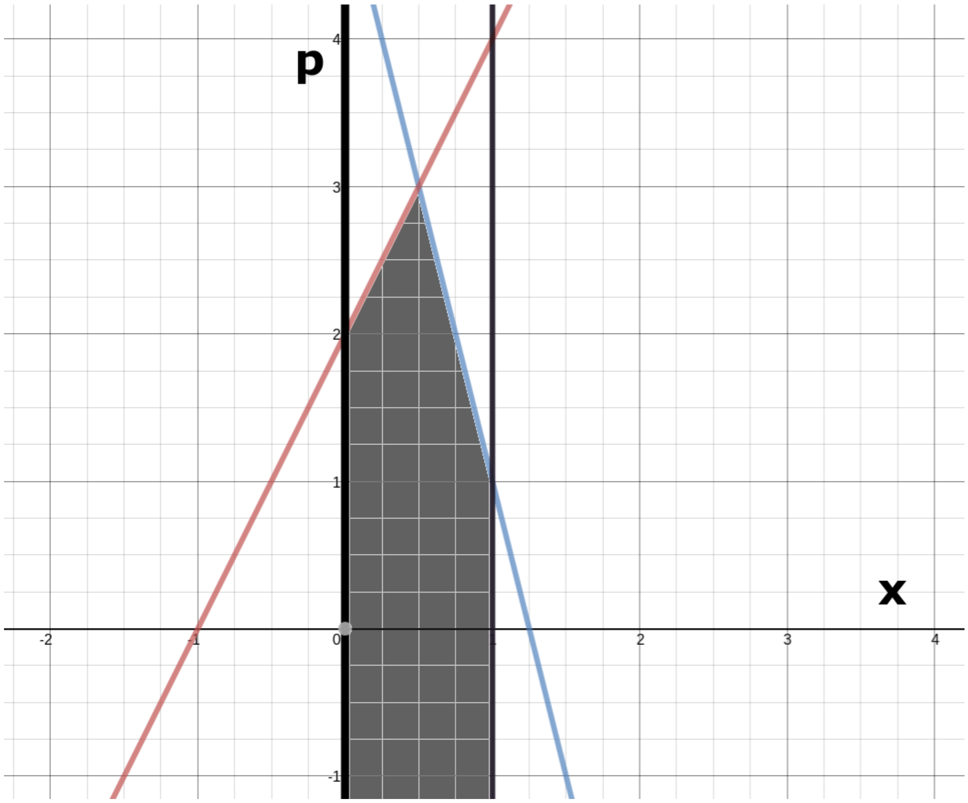
(a) x_i is the probability that Alice plays strategy i and variable p denotes Alice's payoff

(b)

$$\begin{aligned} &\max p \\ &p \leq 4x_1 + 2x_2 \\ &p \leq x_1 + 5x_2 \\ &x_i \geq 0 \\ x_1 + {} &x_2 = 1 \end{aligned}$$

(c)

$$\begin{aligned} &\max \; p \\ &p \leq 2x_1 + 2 \\ &p \leq -4x_1 + 5 \\ &x_1 \geq 0 \\ &x_1 \leq 1 \end{aligned}$$

(d) 

(e)

$$x_1 = \frac{1}{2}$$
$$x_2 = \frac{1}{2}$$
$$p = 3$$

4 Repairing a Flow

In a particular network $G = (V, E)$ whose edges have integer capacities c_e, we have already found a maximum flow f from node s to node t where f_e is an integer for every edge. However, we now find out that one of the capacity values we used was wrong: for edge (u, v) we used c_{uv} whereas it should have been $c_{uv} - 1$. This is unfortunate because the flow f uses that particular edge at full capacity: $f_{uv} = c_{uv}$. We could redo the flow computation from scratch, but there's a faster way.

Describe an algorithm to repair the max-flow in $O(|V| + |E|)$ time. Also give a proof of correctness and runtime justification.

Solution:

Main Idea: In the residual graph with the original capacity for (u, v), use DFS to find a path from t to v and from u to s. Join these paths by adding the edge (v, u) in between them to get a path p. Update f by pushing 1 unit of flow from t to s on p (i.e. for each $(i, j) \in p$, reduce f_{ji} by 1). Finally, use DFS to see if the residual graph of the resulting graph, using the new capacity for (u,v), has an $s-t$ path. If so, push 1 unit of flow on this path.

Proof of Correctness: Consider the residual graph of G. If (u,v) is at capacity, then there is a flow of at least 1 unit going from v to t, and since f_e are integral there is a path from v to t with at least one unit of flow moving through it. So the residual graph will have a path from t to v. Similarly, there will be a path from u to s in the residual graph, so the algorithm can always find p correctly.

Note that the size of the maximum flow in the new network will be either the same or 1 less than the previous maximum flow (because changing one edge can change the capacity of the min-cut by at most 1). In the former case, after pushing flow from t to s on p, it is possible to push more flow from s to t, so there must be an $s-t$ path in the new residual graph, which the algorithm will find and push at least 1 unit of flow on because all capacities and flow values are integral. Thus the algorithm finds the new max flow correctly. In the latter case, we will already have the max flow after pushing flow backwards on p, so our algorithm is still correct.

Runtime: The runtime is $O(|V|+|E|)$ because the algorithm just calls DFS thrice.

5 Generalized Max Flow

Consider the following generalization of the maximum flow problem.

You are given a directed network $G=(V,E)$ where edge e has capacity c_e. Instead of a single (s,t) pair, you are given multiple pairs $(s_1,t_1),...,(s_k,t_k)$, where the s_i are sources of G and t_i are sinks of G. You are also given k (positive) demands $d_1,...,d_k$. The goal is to find k flows $f^{(1)},...,f^{(k)}$ with the following properties:

(a) $f^{(i)}$ is a valid flow from s_i to t_i.

(b) For each edge e, the total flow $f_e^{(1)}+f_e^{(2)}+...+f_e^{(k)}$ does not exceed the capacity c_e.

(c) The size of each flow $f^{(i)}$ is at least the demand d_i.

(d) The size of the *total* flow (the sum of the flows) is as large as possible.

Write a linear problem using the variables $f_e^{(i)}$ whose optimal solution is exactly the solution to this problem. For each constraint as well as the objective in your linear program briefly explain why it is correct. (Note: Since linear programs can be solved in polynomial time, this implies a polynomial-time algorithm for the problem)

Solution: We have variables $f_{(u,v)}^{(i)}$ for all $1 \le i \le k$ and $(u,v) \in E$. The total flow across each edge should be at most the capacity of the edge. Hence, we have the constraint

$$\forall (u,v) \in E \qquad \sum_{i=1}^{k} f_{(u,v)}^{(i)} \le c_{(u,v)}$$

Also, flow conservation for the flow $f^{(i)}$ must hold for all vertices except s_i and t_i. This gives the set of constraints

$$\forall 1 \le i \le k, \forall v \in V \backslash \{s_i,t_i\} \qquad \sum_{(u,v)\in E} f_{(u,v)}^{(i)} = \sum_{(v,w)\in E} f_{(v,w)}^{(i)}$$

Finally, the for each i, $f^{(i)}$ must be greater than the corresponding demand.

$$\forall 1 \leq i \leq k \qquad \sum_{(s_i,u)\in E} f^{(i)}_{(s_i,u)} \geq d_i$$

The objective is simply to maximize the sum of all the flows.

$$\max \sum_{i=1}^{k} \sum_{(s_i,u)\in E} f^{(i)}_{(s_i,u)}$$

6 Reductions Among Flows

Show how to reduce the following variants of Max-Flow to the regular Max-Flow problem, i.e. do the following steps for each variant: Given a directed graph G and the additional variant constraints, show how to construct a directed graph G' such that

(1) If F is a flow in G satisfying the additional constraints, there is a flow F' in G' of the same size,

(2) If F' is a flow in G', then there is a flow F in G satisfying the additional constraints with the same size.

Prove that properties (1) and (2) hold for your graph G'.

(a) **Max-Flow with Vertex Capacities:** In addition to edge capacities, every vertex $v \in G$ has a capacity c_v, and the flow must satisfy $\forall v : \sum_{u:(u,v)\in E} f_{uv} \leq c_v$.

(b) **Max-Flow with Multiple Sources:** There are multiple source nodes $s_1, \ldots, s_k$, and the goal is to maximize the total flow coming out of all of these sources.

Solution:

(a) Split every vertex v into two vertices, v_{in} and v_{out}. For each edge (u, v) with capacity c_{uv} in the original graph, create an edge (u_{out}, v_{in}) with capacity c_{uv}. Finally, if v has capacity c_v, then create an edge (v_{in}, v_{out}) with capacity c_v. If F' is a flow in this graph, then setting $F(u, v) = F'(u_{out}, v_{in})$ gives a flow in the original graph. Moreover, since the only outgoing edge from v_{in} is (v_{in}, v_{out}), and incoming flow must be equal to outgoing flow, there can be at most c_v flow passing through v. Likewise, if F is a flow in the original graph, setting $F'(u_{out}, v_{in}) = F(u, v)$, and $F'(v_{in}, v_{out}) = \sum_u F(u, v)$ gives a flow in G'. One can easily see that these flows have the same size.

(b) Create one "supersource" S with edges (S, s_i) for each s_i, and set the capacity of these edges to be infinite. Then if F is a flow in G, set $F'(S, s_i) = \sum_u F(s_i, u)$. Conversely, if F' is a flow in G', just set $F(u, v) = F'(u, v)$ for $u \neq S$, and just forget about the edges from S. One can easily see that these flows have the same size.

7 A Flowy Metric

Consider an undirected graph G with capacities $c_e \geq 0$ on all edges. G has the property that any cut in G has capacity at least 1. For example, a graph with a capacity of 1 on all edges is connected if and only if all cuts have capacity at least 1. However, c_e can be an arbitrary nonnegative number in general.

1. Show that for any two vertices $s, t \in G$, the max flow from s to t is at least 1.

2. Define the *length* of a flow f to be $\texttt{length}(f) = \sum_{e \in G} |f_e|$. Define the *flow distance* $d_{\texttt{flow}}(s,t)$ to be the minimum length of any $s-t$ flow f that sends one unit of flow from s to t and satisfies all capacities; i.e. $|f_e| \leq c_e$ for all edges e.

   Show that if $c_e = 1$ for all edges e in G, then $d_{\texttt{flow}}(s,t)$ is the length of the shortest path in G from s to t.

   (*Hint*: Let $d(s,t)$ be the length of the shortest path from s to t. A good place to start might be to first try to show $d_{\texttt{flow}}(s,t) \leq d(s,t)$. Then try to show $d_{\texttt{flow}}(s,t) \geq d(s,t)$)

3. **(Optional)** The shortest path satisfies the *triangle inequality*, that is for three vertices, s, t, and u in G, if $d(x,y)$ is the length of the shortest path from x to y, then $d(s,t) \leq d(s,u) + d(u,t)$. Show that the triangle inequality also holds for the flow distance. That is; show that for any three vertices $s, t, u \in G$

$$d_{\texttt{flow}}(s,t) \leq d_{\texttt{flow}}(s,u) + d_{\texttt{flow}}(u,t)$$

   even when the capacities are arbitrary nonnegative numbers.

Solution:

1. For any two vertices s, t, the min cut separating s and t has value at least 1. Therefore, by the max flow-min cut theorem, the maximum $s-t$ flow has value at least 1.

2. First, we show that $d_{\texttt{flow}}(s,t) \leq d(s,t)$ for all $s, t \in G$. Let f be the unit $s-t$ flow that sends one unit of flow along the shortest path from s to t. Then $\texttt{length}(f)$ is the length of this path, which is $d(s,t)$. Since $d_{\texttt{flow}}(s,t)$ is the minimum length of any unit $s-t$ flow, $d_{\texttt{flow}}(s,t) \leq \texttt{length}(f) = d(s,t)$.

   Now, we show that $d(s,t) \leq d_{\texttt{flow}}(s,t)$. Consider the minimizing flow; that is the unit $s-t$ flow f with length $d_{\texttt{flow}}(s,t)$. Any unit $s-t$ flow f can be written as

$$f = \sum_{\text{s-t paths } p \text{ in } G} w_p f_p$$

   where f_p is the unit flow along the path p, $\sum_{\text{s-t paths } p} w_p = 1$, $w_p \geq 0$ for all $s-t$ paths p, and all paths go in the same direction from s to t through any edge e. In other words, we can argue that a flow f will send some (potentially zero) fraction of its flow through each possible $s-t$ path. For example, if there are exactly 2 paths from s to t, we could send exactly half of our flow along each path, or .3 along one path and .7 along the other. Notice this is true regardless of whether or not these paths are disjoint.

This means that

$$\texttt{length}(f) = \sum_{\text{s-t paths } p \text{ in } G} w_p \texttt{length}(f_p)$$

i.e. $\texttt{length}(f)$ is the average of the lengths of many $s-t$ paths in G. In particular, there exists an $s-t$ path q in G with $\texttt{length}(f_q) \leq \texttt{length}(f) = d_{\texttt{flow}}(s,t)$. $d(s,t) \leq \texttt{length}(f_q)$, finishing the proof of the desired inequality.

3. Let f and g be the $s-u$ and $u-t$ flows with lengths $d_{\texttt{flow}}(s,u)$ and $d_{\texttt{flow}}(u,t)$ respectively. We wish we could define an $s-t$ flow $r = f+g$ and use this flow to show that $d_{\texttt{flow}}(s,t) \leq \texttt{length}(f) + \texttt{length}(g)$. $f+g$ does not have to respect the capacities c, so it is not feasible. This is strange, though, because f and g individually do satisfy the capacities c. In order for $f+g$ to violate an edge capacity, flow from both f and g must cross the violated edge. But this implies that the $f+g$ flow would form a cycle, which is wasteful. We want to remove these cycles.

   We now formalize this intuition using max flow-min cut duality. Define a new capacity vector c' with $c'_e = \max(|f_e|, |g_e|)$. Notice that $c'_e \leq c_e$ for all edges e because f and g respect G's capacities, so any c'-respecting flow is also a c-respecting flow. Futhermore, any c'-respecting flow h has length at most

$$\texttt{length}(h) \leq \sum_e c'_e \leq \sum_e |f_e| + |g_e| = \texttt{length}(f) + \texttt{length}(g) = d_{\texttt{flow}}(s,u) + d_{\texttt{flow}}(u,t)$$

   Therefore, it suffices to show that there is a unit $s-t$ c'-respecting flow. By max flow-min cut, it suffices to show that all $s-t$ cuts have capacity at least 1. Consider any set of vertices S containing s but not t. If $u \notin S$, then $1 \leq \sum_{e\in\partial S} |f_e| \leq \sum_{e\in\partial S} c'_e$. If $u \in S$, then $1 \leq \sum_{e\in\partial S} |g_e| \leq \sum_{e\in\partial S} c'_e$. In particular, the cut defined by S always has capacity at least 1. Therefore, the min $s-t$ c'-cut has capacity at least 1. Therefore, there exists a unit $s-t$ flow with length at most $d_{\texttt{flow}}(s,u) + d_{\texttt{flow}}(u,t)$, as desired.