Homework 8: pencil.io

Due: Nov. 27, 2018 at 6:00 PM (early)
Nov. 30, 2018 at 6:00 PM (on time)
Dec. 2, 2018 at 6:00 PM (late)

This is a partner homework as usual: work on all problems together; CC your partner on all emails to the TA list; you are responsible for everything you and your partner submit and it is an academic code violation to submit something that is not yours. Check for your partner in the Google doc that was sent out. **If you don't have a partner, please let us know.**

Each of the 5 problems in this assignment may be turned in separately, for a separate deadline. Run `cs157_handin hw8-p3` to copy everything from your current directory to our grading system as a submission for problem 3.

Make sure that you clearly indicate on each problem whihc track you're in (either **1-credit** or **full**). **1-credit track:** do problems 1, 4, and 5 *and indicate "***1-credit track***" on each problem.*

Problem 1

(20 points) We are pivoting our product to better leverage the synergies that we have created and we need to decide how to market our latest disruption—it's like Uber, but for pencils. (We have big expansion plans for pen market penetration but have not hired developers for this yet.)

You are in charge of our marketing strategy: each week you can choose to create viral marketing videos advertising either the low *price* of our pencils, or their high *quality*—designed with quantum annealing technology! Meanwhile, our main competitor, Office Depot, must decide how to set the prices for their own pencils: *low* (L), *medium* (M), or *high* (H). Once both parties have chosen an option, our profits are determined by the following table (we do not care what Office Depot's profits are):

	L	M	H
P	\$0	\$3000	\$4000
Q	\$3000	\$2000	\$2000

Namely, if we choose to advertise based on how low our prices are, but Office Depot secretly chooses to sell pencils at lower prices, then we will make \$0, while if Office Depot chooses medium prices we will make \$3000 and if Office Depot chooses high prices we will make \$4000. On the other hand, if we choose to emphasize the quantum annealed pencil design in our advertising, then if Office Depot chooses low prices, the tech-buzzword fanatics will suspect our pencils are better than the No. 2 pencils that Office Depot carries and reward us with \$3000 profit; while if Office Depot chooses medium or high prices, consumers will suspect that we are an IPO scam and we will make only \$2000.

We do not know what prices Office Depot will pick, so we want to choose a *probability distribution* over our strategies that will guarantee as much money as possible *no matter what Office Depot does*. Namely, suppose we advertise based on price with probability p and advertise based on quality quantum technology with probability q, then the expected profit if Office Depot chooses

"H" (high prices) is $4000p + 2000q$. Your task is to choose probabilities p, q such that $p + q = 1$, and such that the *minimum* of the values of the three columns L, M, H is as *high* as possible.

To do this, design a linear program with three variables: p, q, and v, where v will represent the expected value (profit) corresponding to probabilities p, q (under the worst case column); your linear program will have 4 constraints: 3 constraints enforcing that v is less than or equal to the value of each of the 3 columns L, M, H, and a final constraint that $p + q = 1$. To complete the linear program, we insist that $p, q \geq 0$, while v is unrestricted (neither restricted to be larger nor smaller than 0), and we ask to maximize v.

1. (5 points) Fill in the details of this linear program, and describe them here. Use Matlab's `linprog` routine to solve your linear program: what are the probabilities p and q, and what is the profit v our start-up can expect to make?

   Important: Before you optimize your linear programs in this problem, run the command `format shortg` to change Matlab's output format to output 5 digits of precision (instead of the default which will swallow small numbers when they are part of vectors with larger numbers).

2. (4 points) Take the dual of this linear program. Use Table 4.1 on page 141 of `http://web.mit.edu/15.053/www/AMP-Chapter-04.pdf` (also linked under "duality" from the "keynots" (classes) section of the course homepage) for details of how to translate each portion of the linear program to its dual form. Your dual linear program should have 4 variables, one for each of the original 4 constraints, and we may as well label these variables by the names of the corresponding constraints, ℓ, m, h for the variables corresponding to the low, medium, and high constraints, and a 4th variable w corresponding to the "probability weight" constraint (though this variable will have a different meaning in the dual). Explicitly describe the dual linear program, and then solve it with Matlab, reporting the values of ℓ, m, h, w at optimum, as well as the objective function value (which should be the same as for the primal).

   If you did this right, the dual to the linear program should look exactly like the game from *Office Depot's* perspective: you sell pencils and can either choose low, medium, or high prices, and meanwhile your adversary will secretly choose an advertising campaign. You want to minimize your opponent's profits, since every dollar spent on your opponent's platform is a dollar not spent at Office Depot.

3. (5 points) The dual solution above should let you write down a pencil-and-paper *proof* that the probabilistic strategy p, q that we originally found for our start-up produces the optimum profit. (Remember from class: the values of variables in the dual solution can be used as multipliers on the constraints from the original linear programming problem, that when combined, yield a proof of optimality.) Prove that the value of v that you found in part 1 is optimal, via this method.

4. (2 points) The dual solution should have two nonzero probabilities and one probability equal to 0, indicating that 2 of the 3 strategies for Office Depot look reasonable, while 1 strategy will never be played. This corresponds to the fact that in the original linear program, 2 of the 3 constraints corresponding to L, M, H are *tight* (the inequalities are at equality), while one is slack. Explain how both the original and dual linear programs identify the same one of the L, M, H strategies as worthless. (The fact that dual variables are nonzero only when primal constraints are at equality is known as "complementary slackness", and is a general principle of linear programming.)

5. (4 points) Find the expected *value* of each of the 5 strategies P, Q, L, M, H of the game, assuming both players play the probability distributions you found above. Show that for all strategies with positive probability, the values are identical, and greater than or equal to the non-played strategies. This shows that we have found a *Nash Equilibrium* of the game: a pair of randomized strategies such that neither player has incentive to deviate.

(For more details on games like this, and how linear programming solves them, see `https://en.wikipedia.org/wiki/Zero-sum_game`.)

Problem 2

(20 points) Many disruptable industries have to solve some version of the following "hiring problem." Suppose you run a start-up business that needs 8000 hours of labor in May, 9000 hours in June, 7000 in July, 10,000 in August, 8000 in September, and and 11,000 in October. Also, suppose that currently, as May starts, you have 60 experienced employees working for you. Each month, each experienced employee can *either* work up to 150 hours in that month *or* work up to 50 hours that month while also training one new hire who will then be ready to work the *following* month (and will be called an "experienced" worker the following month). At the end of each month, 10% of your experienced employees quit. It costs $3400 a month to employ an experienced worker, and $1800 a month for each trainee.

1. (6 points) Argue that the set of feasible hiring strategies is convex: if two different strategies are both feasible, then their average is feasible too. Thinking precisely about this should help get you started on the next part of the problem. (Start by figuring out what a natural set of variables for the problem is, then consider two feasible vectors of variables and start thinking about all the constraints that apply.)

2. (7 points) Write a linear program that represents this problem, and describe what corresponds to what, and why it accurately represents the problem. (Note, do not worry about integrality of the solution; the optimum of the linear program may include things that mean, for example, "hire 27.3 people in June".)

3. (7 points) Solve the linear program via Matlab's `linprog` routine, and describe the optimal hiring strategy and its cost, along with the details of how you set up the problem in Matlab. (Type `help linprog` for details.)

Problem 3

(20 points) Linear regression is a fundamental problem: given n points (x_i, y_i), find a line that best approximates them. While you may have seen "least squares" regression, which finds a line $y = p + q \cdot x$ that minimizes the sum of the *squares* of the distances to the points, in this problem we will consider two variants, both using the absolute value of the distance (instead of its square).

1. (5 points) Consider the problem of finding p, q that minimize the sum of the absolute values of the (vertical) distance of the line $y = p \cdot x + q$ to each of a series of n points (x_i, y_i). Namely, find p, q that minimize $\sum_{i=1}^{n} |px_i + q - y_i|$. The absolute value is often referred to as the "L1

metric", so this task is often called "L1 fitting" or "L1 regression" as opposed to the more standard "L2 regression".

Fill in the template `L1Regression` that takes as input vectors x and y of equal length and outputs the specification of a linear program, as could be input to Matlab's `linprog` routine. The **first** and **second** entries in a solution to this linear program must encode optimal values of p and q respectively.

(**Hint:** You cannot express absolute value constraints exactly in the linear programming framework, but in certain cases you can do something just as good by using constraints and adding a new term to the objective function. Remember, the feasible region of a linear program is *convex*, and, essentially, any convex region can be expressed via a linear program. In two dimensions, the set of points (x, y) for which $y = |x|$ is *not* convex. But the set for which $y \geq |x|$ is convex. Start here.)

2. (5 points) Repeat the above task for the related problem of finding the line $y = p \cdot x + q$ that minimizes the *maximum absolute deviation* from the line. Namely, given a series of n points (x_i, y_i), find p, q that minimizes: $\max_i |px_i + q - y_i|$. Fill in the template `L1MaxRegression` as above to produce a linear program to solve this problem.

3. (6 points) Show that *each* of the previous two problems is convex in the following sense:

   (a) If two different lines (p_1, q_1) and (p_2, q_2) each have total distance to the points (x_i, y_i) smaller than some v, then their average $(\frac{p_1+p_2}{2}, \frac{q_1+q_2}{2})$ also specifies a line that has total distance to the points smaller than v.

   (b) If two different lines (p_1, q_1) and (p_2, q_2) each have max distance to the points (x_i, y_i) smaller than some v, then their average $(\frac{p_1+p_2}{2}, \frac{q_1+q_2}{2})$ also specifies a line that has max distance to the points smaller than v.

4. (4 points) Run your two L1 regression routines on data x, y that you generate, and also try Matlab's built-in least-squares regression routine `polyfit(x,y,1)`. For example, if `x=1:10` and `y=2*x+1`, then the first two entries returned in all three cases should be $(2, 1)$. However, for this part, come up with an example where the three regression routines return different answers, and further, where the built-in least squares regression looks *worse* than the result of one of the two new variants you constructed. Plot these three different "best fit lines" along with the original data on a graph that you save and turn in. Write a couple sentences explaining why least-squares regression does not work well in the case you found.

   (**Note:** The easiest way to plot multiple things in Matlab is with the `hold on` command, which tells Matlab not to erase what is already plotted—`hold off` reverts this. Type `help plot` to see options for plotting scatter plots, or different colors, etc. Type `help axis` to see options for adjusting the plot window, if necessary.)

Problem 4

(20 points) Some courses are rewarding to take; other courses you take only because they are prerequisites for rewarding courses later on.

Suppose a university offers n courses, and each course i has a set of prerequisites $p_i \subset [n]$, all of which you must take if you plan on taking course i. (You may assume that there are no cyclic dependencies in the prerequisite list.) In addition, for each course i, you also know how rewarding

it will be to take, a number r_i, in arbitrary units, which could be positive or negative! Your goal is to find a set C of courses to take, subject to the prerequisite constraints (for each course i and each prerequisite $j \in p_i$, if i is in C then j is in C), in order to maximize $\sum_{i \in C} r_i$.

1. (6 points) Express this problem as an integer program, where each course corresponds to a variable that has value 0 if the course is not taken, and 1 if it is. Each constraint should enforce a **single** relation of the form "course j is a prerequisite for course i".

2. (14 points) Show that this problem can in fact be solved in polynomial time by linear programming: taking the above integer program and ignoring the integrality constraints, show that the resulting linear program has an optimal solution all of whose coordinates are integers.

   (**Hint:** Two possible ways to prove this include 1) consider a hypothetical optimum solution to the linear program, take all its non-integer coordinates, and figure out how to modify them to a solution that is at least as good but which has more of its coordinates equal to 0 or 1; or 2) make use of the fact that the set of optima of a linear program always includes a vertex of the constraint polytope, and show that such a vertex must have 0-1 coordinates.)

Problem 5

Convexity:

(20 points) In this problem we will be exploring various aspects of convexity. Recall the definition of a convex function (`http://en.wikipedia.org/wiki/Convex_function`): a function f from n-dimensional space to the real numbers is convex if for any two inputs x and y, the line segment in the graph of f from $(x, f(x))$ to $(y, f(y))$ lies on or above the graph of f. (The picture is easiest to draw for dimension $n = 1$.) To express this slightly more formally, for any interpolation parameter λ between 0 and 1, we have the condition $\lambda f(x) + (1 - \lambda) f(y) \geq f(\lambda x + (1 - \lambda) y)$. Make sure you understand what this means in one dimension before moving on.

1. (5 points) Our rockstar dev just pushed out the *golden section search* algorithm, `http://en.wikipedia.org/wiki/Golden_section_search`. It is an algorithm for optimizing convex functions on a line that takes logarithmic time just like binary search, but is slightly faster, and with other nice properties (maybe we can sell it with an as-a-service model...).

   Review the Wikipedia article to familiarize yourself with it. In this problem we will extend this algorithm to higher dimensions, so make sure you are comfortable with the basic idea first. An example implementation of this algorithm is in the stencils directory (play around with the code if you find it helpful, but there is no need to code for this problem).

   At each moment in the golden section search algorithm, it considers the function value at three unequally spaced points, $x_1 < x_2 < x_3$, where x_2 is ϕ times closer to one endpoint than than the other, where ϕ, the golden ratio, is $\frac{\sqrt{5}+1}{2} \approx 1.618$. Consider the case where x_2 is closer to x_1 than x_3. By the induction hypothesis of the algorithm, $f(x_2)$ is less than or equal to $f(x_1)$ and $f(x_3)$.

   Clearly the minimum value of f is at most $f(x_2)$. Find the best *lower bound* for f, in terms of $f(x_1)$, $f(x_2)$, and $f(x_3)$, given that f is convex. Explain.

   (**Hint:** as you are trying to build intuition and simplify expressions in this problem, it may help to be aware of the many nice relations between powers of ϕ, which is equivalently defined as the solution to the equation $\phi^2 = \phi + 1$, including $\frac{1}{\phi} = \phi - 1$, and $\frac{1}{\phi^2} = 2 - \phi = 1 - \frac{1}{\phi}$.)

2. (3 points) Suppose we want our golden section search algorithm to return a value x such that $f(x)$ is within ϵ of the global minimum. What *stopping condition* for the algorithm does your results from the previous part suggest? Explain.

3. (6 points) We generalize this to higher dimensions by recursively reducing to lower–dimensional problems. Assume (this would be an induction hypothesis) that you have an algorithm that can minimize arbitrary convex functions in $n-1$ dimensions. Then we can minimize an n–dimensional function f as follows: Given a value y for the last coordinate of the input, define $f_y(x)$ to be the function that takes an $n-1$-dimensional input x, and evaluates f on the concatenation (x, y); for any given y, our induction hypothesis lets us minimize f_y, since this is a $n-1$-dimensional convex optimization problem; define $g(y)$, to be the result of optimizing f_y; now, $g(y)$ is a one–dimensional function, (evaluated via our inductive hypothesis), and, if $g(y)$ is convex, we can optimize it via golden section search (using recursive calls to the $n-1$–dimensional algorithm for each function evaluation).

   For this framework to work, we need to show that $g(y)$ is convex. The heart of the proof is the two dimensional case:

   Given a convex function in two dimensions, $f(x, y)$, if we define a new function $g(x) = \min_y f(x, y)$ to be the minimum of f along its second dimension, for each value of x, then g is a convex function. Prove this.

4. (6 points) A further complication with this kind of recursive optimization procedure has to do with numerical precision. The golden section search algorithm will not find the *exact* minimum value of its input function, but will, rather, return a value which may be larger than the minimum. This error may make subsequent layers of the recursion have even larger errors.

   Suppose that you are trying to minimize a convex function g, but that you only have *approximate* access to g: you can evaluate a function $\widetilde{g}$ that, when evaluated at an input x is guaranteed to return a value in the interval $[g(x), \epsilon + g(x)]$, for some error parameter ϵ. Recall that the golden section search algorithm when run on $\widetilde{g}$, in each iteration compares values $\widetilde{g}(x_2)$ and $\widetilde{g}(x_4)$ to decide whether to continue the search in the left or the right. The issue is that it might be the case that $\widetilde{g}(x_2) < \widetilde{g}(x_4)$ while $g(x_2) > g(x_4)$, in which case the algorithm will go in the *wrong direction*. Show that the **first** time the algorithm makes a bad decision, recursing to an interval that does *not* contain the global minimum, the true global minimum of g is at least $\widetilde{g}(x_2) - \phi^2\epsilon$.

   (Putting together all the pieces you have proven lets us design a recursive algorithm for convex minimization, of the flavor of the recursive binary search algorithm we saw in class—though with a provably robust error analysis. This algorithm uses golden section search along each dimension, and where, i dimensions "down" in the recursion, the error parameter ϵ in your algorithm of part 2 should be a ϕ^{2i} factor smaller than the overall accuracy goal of the algorithm. Since each recursive layer of this algorithm makes a logarithmic number of calls to the next layer, the total time is not logarithmic, but, in n dimensions, the nth power of the logarithm of the desired accuracy. For large n this is very bad, and we would use the ellipsoid algorithm for convex optimization instead.)