Homework 8

Due: Apr 13, 2013 2:00 PM (early)
Apr 15, 2013 11:59 PM (on time)
Apr 17, 2013 11:59 PM (late)

This is a pair assignment, with all of the same rules as usual.

Problems 2 and 6 should be handed in electronically, by running `cs157_handin hw8-p2` or `cs157_handin hw8-p6`, and all the other problems should be handed in on paper.

Problem 1

Many industries have to solve some version of the following "hiring problem." Suppose you run a business that needs 8000 hours of labor in May, 9000 hours in June, 7000 in July, 10,000 in August, 9000 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 for 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. (10 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".)

2. (5 points) Solve the linear program via Matlab's `linprog` routine, and describe the optimal hiring strategy, along with the details of how you set up the problem in Matlab. (Type `help linprog` for details.)

Problem 2

Linear regression is a fundamental problem: given n points (x_i, y_i), find a line that best approximates them. While you have probably 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. (10 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 seconds entries in a solution to this linear program must encode optimal values of p and q respectively.

2. (7 points) Repeat the above task for the related problem of finding the line $y = p \cdot x + q$ that minimizes the *maximum 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. (3 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.

   (**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 3

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. (5 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. (10 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 4

Convexity:

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) Recall the *golden section search* algorithm from the optimization lab (`http://en.wikipedia.org/wiki/Golden_section_search`), for minimizing convex functions f in one dimension. At each moment in the algorithm, you will be considering 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 $\frac{\sqrt{5}+1}{2}$. Consider the case where x_2 is closer to x_1 than x_3. Also, 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 on this interval 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.

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 do your results from the previous part suggest?

3. (6 points) In class, Josh Freilich made an interesting proposal for how to generalize algorithms like this to higher dimensions: assume you have an algorithm that can minimize convex functions in $n-1$ dimensions; then we can solve the full n-dimensional minimization algorithm by just running a 1-dimensional minimization algorithm over the function that is the result of minimizing the input function over its remaining $n-1$ dimensions (using the $n-1$-dimensional algorithm as a black box). For this algorithm to be reasonable, we need one crucial fact:

   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 f, but that you only have *approximate* access to f: you can evaluate a function $\widetilde{f}$ that, when evaluated at an input x is guaranteed to return a value in the interval $[f(x), \epsilon + f(x)]$, for some error parameter ϵ. Recall that the golden section search algorithm when run on $\widetilde{f}$, in each iteration compares values $\widetilde{f}(x_2)$ and $\widetilde{f}(x_4)$ to decide whether to continue the search in the left or the right. The issue is that it might be the case that $\widetilde{f}(x_2) < \widetilde{f}(x_4)$ while $f(x_2) > f(x_4)$, in which case the algorithm will go in the *wrong direction*. Show that the first time this happens, the true global minimum of f is at least $\widetilde{f}(x_2) - \phi^2\epsilon$.

   (**Hint:** The reason for using $\phi = \frac{\sqrt{5}+1}{2}$ in this algorithm is that this value of ϕ satisfies $1 + \phi = \phi^2$, or equivalently that $\frac{1}{\phi} = \phi - 1$. You will need relations like this to simplify your calculations.)

(Putting together all the pieces you have proven lets us design a recursive algorithm that 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.)

Problem 5

Duality:

1. (10 points) Consider the linear program $\min_x u \cdot x$ subject to the constraints $A \cdot x \geq b$, where the vector u represents "up". Thus we are trying to find the "lowest" point in the feasible region. Note that we do *not* have the usual constraints "$x \geq 0$"! Assume for this problem that this linear program has feasible solutions.

   For the sake of concreteness, assume we have n variables and m constraints. That is, A is a matrix with n columns and m rows. We denote the ith row of A by A_i, and the ith entry of b by b_i. Thus the ith constraint of the problem is: $A_i \cdot x \leq b_i$.

   Consider the following "physics thought experiment": drop a small marble inside the feasible region, and wait for it to settle to the lowest point. This process "solves" the linear program. Let x^* denote the solution to the linear program, the location of the marble. The marble is now resting at the bottom of the feasible region, supported by some of the constraints, and in particular, by constraints that pass through x^*. We appeal to the following physics intuition: 1) the force that constraint i exerts on the marble must be a nonnegative multiple of A_i, where A_i is the ith row of the matrix A, associated with the ith constraint $A_i \cdot x \geq b_i$; 2) the total force these constraints exert on the marble must be in the up direction, u; 3) only constraints that pass through x^* can exert force on the marble.

   We express this physics intuition as the following fact, which you do not need to prove: 1) there are nonnegative values $y_1, \ldots, y_m$ such that 2) $y_1 A_1 + \ldots + y_m A_m = u$, and where 3) $y_i = 0$ if constraint i does not pass through x^* (explicitly, constraint i, which says $A_i \cdot x \leq b_i$, passes through x^* if $A_i \cdot x^* = b_i$).

   Prove the linear programming duality theorem in this setting!

   Namely, recall from lecture that the dual to our linear program is: $\max_y b^T \cdot y$ subject to the constraints $A^T \cdot y = u$ and $y \geq 0$ (the unbounded variables x correspond to equality constraints under duality). Show that the coefficients $\{y_i\}$—that we found in our physics thought experiment above—make everything work out: 1) show that $\{y_i\}$ is a feasible solution to the dual linear program; 2) show that the objective value of $\{y_i\}$ in the dual linear program *equals* the optimal objective value of the original linear program, $u \cdot x^*$; 3) summarize what you have found.

2. A *two-player zero-sum game* is specified by a matrix A, where one player picks a row i of the matrix (this player is called the "row player"), and the other player picks a column j of the matrix; the element $A(i, j)$ specified by this row and column is the *result* of the game, and the row player gets $A(i, j)$ dollars (which may be positive or negative), while the column player

gets $-A(i,j)$ dollars. For example, the familiar game rock-paper-scissors has the matrix

$$A = \begin{array}{c|ccc} & r & p & s \\ \hline r & 0 & -1 & 1 \\ p & 1 & 0 & -1 \\ s & -1 & 1 & 0 \end{array}$$

We consider *randomized* strategies for the players. Suppose that you, the row player, know that the column player picks rock with probability 0.5, paper with probability 0.2, and scissors with probability 0.3, then your *expected payoff* for each of your three strategies may be found by taking 0.5 times the first column of A, plus 0.2 times the second column of A, plus 0.3 times the third column of A, which equals $\begin{bmatrix} 0.1 \\ 0.2 \\ -0.3 \end{bmatrix}$, meaning that you could make expected profit of 0.1 by choosing rock all the time, 0.2 by choosing paper all the time, or -0.3 by choosing scissors all the time. To express this slightly more in the language of linear programming, the entry 0.1 guarantees you an expected profit of at least 0.1, regardless of the other entries of the matrix, etc. Given these choices, you will presumably choose paper, for an expected profit of 0.2. Your opponent wants to minimize your profit.

(a) (7 points) Write a linear program that your opponent could solve that would *minimize* your expected profit, assuming you respond to his probability distribution intelligently, as we just saw. Explain how it works.

(b) (3 points) Take the dual of this linear program, and explain how it solves the same problem as above but with the roles of the two players swapped.

(c) (5 points) Because these two linear programs are dual, the linear programming duality theorem says that they have the *same* optimal objective value. Consider the following process: you solve the linear program from part b) and choose a row according to that probability distribution, and your opponent solves the linear program from part a) and chooses a column according to that probability distribution. Conclude that both players would be happy to play like this, that neither can make more profit by changing her strategy.

(This pair of probabilistic strategies is called a *Nash equilibrium*, and we have shown its existence here, and found it, by the power of linear programming duality.)

Problem 6

(5 points)

As you may have noticed, there are sections of the course where we do not know of suitable suggested reading materials. Most recently, homework 7 (optimization) was not associated with any reading. For the 5 points here, please suggest one or two readings that would be helpful for the class next year, either on the topic of optimization, or on any of the topics that you found confusing or insufficiently covered (Fourier transforms? "Map of the computer"?). Turn in a text file describing or linking these documents, along with a brief explanation of what you appreciate about them. Your grade here will be proportional to how much we appreciate what you turned in. More points will be given for readings that would most improve the course, and that are suggested by few other people.