COS 521: Advanced Algorithms
Homework 2

Zachary Hervieux-Moore

Friday 21st October, 2016

Exercise 1: In class we saw a hash to estimate the size of a set. Change it to estimate frequencies. Thus there is a stream of packets each containing a *key* and you wish to maintain a data structure which allows us to give an estimate at the end of the *number of times* each key appeared in the stream. The size of the data structure should not depend upon the number of distinct keys in the stream but can depend can depend upon the success probability, approximation error, etc. Just shoot for the following kind of approximation: if a_k is the true number of time that key k appeared in the stream then your estimate should be $a_k \pm \epsilon(\sum_k a_k)$. In other words, the estimate is going to be accurate only for keys that appear frequently ("heavy hitters") in the stream. (This is useful in detecting anomalies or malicious attacks.) Hint: Think in terms of maintaining $m_1 \times m_2$ counts using as many independent hash functions, where each key updates m_2 of them.

Answer: As the hint suggests, we will maintain a $m_1 \times m_2$ counts where we have m_2 pairwise independent hash functions $(h_1(k), \ldots, h_{m_2}(k))$ that have m_1 possible buckets. Then we hash the key k as follows:

$$hash(k) = ((1, h_1(k), (2, h_2(k)), \ldots, (m_2, h_{m_2}(k))))$$

That is, we generate a hash for one cell in each row of our $m_1 \times m_2$ array. For every cell, increment it by 1. Then, to get the frequency, we simply report the minimum of these m_2 cells. Note that this is an upper bound since other keys can hash to these cells. Let k be the key we want to extract its frequency. Note that any other key has a probability of $\frac{1}{m_1}$ to collide with the minimum bucket. Thus, the expected value of the reported a_k is,

$$a_k + \frac{1}{m_1} \sum_{j \neq k} a_j$$

Hence, we can pick m_1 sufficiently large such that the above sum is greater than $a_k + \epsilon \sum_{j \neq k} a_j$ with probability at most 1/2. Note that this doesn't depend on the number of distinct keys, but only on our choice of ϵ. Also, the summation is a sum of Bernoulli random variables with probability $\frac{1}{m_1}$ multiplied by a_j. Thus, it is easy to see from this fact that m_1 is going to depend on ϵ like $\frac{1}{\epsilon^n}$ for some n.

Consider the worst case where all the hashes exceed this bound. This is going to occur with a probability at most $(1/2)^{m_2}$. Hence, we pick m_2 sufficiently large to achieve the desired rate of success.

Exercise 2: (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 to solve arbitrary linear programs to obtain approximate solutions.

$$\begin{aligned} Ax &\geq b \\ x &\geq 0 \\ \sum_{i=1}^{n} x_i &= 1 \end{aligned}$$

1) (Duality) Show a simple method to solve the following linear program for any weights $w_1, w_2, \ldots, w_m$

$$\begin{gathered} \max \sum_{j=1}^{m} w_j (a_j^T x - b_j) \\ x_i \geq 0 \quad \forall\, i \in 1, 2, \ldots, n \\ \sum_{i=1}^{n} x_i = 1 \end{gathered}$$

   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 primal LP is infeasible.

2) The above setting of finding weights that certify the infeasibility of the primal might remind you of the setting of weighting the experts via multiplicative weights update rule discussed in class. Use these idaes to obtain an algorithm that a) either finds a set of non-negative weights certifying infeasibility of the primal or b) finds a solution x that approximately satisfies all the contraints of the primal, i.e., for each $1 \leq i \leq m$, $Ax - b \geq -\epsilon$, $x \geq 0$ and $\sum_{i=1}^{n} x_i = 1$. Give a bound, as tight as possible, for the number of update steps required 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 the primal and maintain a weighting of experts (starting with the uniform weighting of all 1's, say) for times $t = 0, 1, \ldots,$ - these are progressively improving guess for the weights. Solve the dual using the weights at time t. If the value of the dual is negative, you are done, otherwise think of the "cost" of the j^{th} expert as $a_j^T x^{(t)} - b_j$ where $x^{(t)}$ is the solution to the dual LP at time t and update the weights.)

Answer:

1) Expand out the maximization,

$$\max \sum_{j=1}^{m} w_j a_j^T x - w_j b_j$$

Since the b_j term has no x dependence, we can just ignore it. Note that this is linear, so it will have the form,

$$\max \sum_{i=1}^{n} \alpha_i x_i$$

Now it is clear to maximize, just set $x_i = 1$ for the largest α_i.

If all the weight $w_1, w_2, \ldots, w_m$ are non-negative yet the value of the program is negative, then this implies that at least one $a_j^T x - b_j < 0 \iff a_j^T x < b_j$ but this breaks one the constraints of the primal LP. Thus, it is infeasible.

2) Following the hint, we identify m "experts", one for each inequality. Thus, we have weights that change over time with the following formula, $w_j^{(t+1)} = (1 - \eta m_j^{(t)}) w_j^{(t)}$ for $i \in 1, 2, \ldots, m$. The costs here are the $m_j^{(t)} = a_j^T x^{(t)} - b_j$. Where $x^{(t)}$ is the solution to the dual primal with weights $w_i^{(t)}$. Thus, if the cost is postive, then it satisfied the constraint of the primal (which is bad for finding infeasibility). So we penalize these experts and favour the experts that have negative costs (since they make the primal infeasible). First, to satisfy the conditions of the Multiplicative Weights Theorem, we rescale all the costs to be in $[-1, 1]$ at each time t. Now we apply the theorem,

$$0 \leq \sum_{t=1}^{T} a_j^T x^{(t)} - b_j + \eta(\sum_{t=1}^{T} |a_j^T x^{(t)} - b_j|) + \frac{\log m}{\eta}$$

Note that $\sum_{t=1}^{T} |a_j^T x^{(t)} - w_j b_j| \leq T$ since the costs are bounded by 1. We now define $\bar{X} = \frac{1}{T} \sum_{t=1}^{T} x^{(t)}$. Since the inequality holds for all t, it work for the average. The inequality becomes,

$$0 \leq \sum_{t=1}^{T} a_j^T \bar{X} - b_j + \eta T + \frac{\log m}{\eta}$$

We set $\eta = \sqrt{\log m/T}$, then $\eta T = \sqrt{T \log m}$, pick $\epsilon = \sqrt{\log m/T}$, so

$$-\epsilon \leq \sum_{t=1}^{T} a_j^T \bar{X} - b_j + \sqrt{T \log m}$$

Since we are using $\bar{X}$, we want $-\epsilon < \sqrt{T \log m}$ per step, or $\epsilon < \sqrt{\log m/T}$. Thus, pick $T > \log m/\epsilon^2$ and we have the guarantee desired.

Exercise 3: In ℓ_2 regression you are given data points $x_1, x_2, \ldots, x_n \in \mathbb{R}^k$ and some values $y_1, y_2, \ldots, y_n \in \mathbb{R}$ and wish to find the "best" linear function that fits this dataset. A frequent chocie for best fit is the one with *least squared error*, i.e. find $a \in \mathbb{R}^k$ 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 points you may need a certain matrix to be invertible, which you can assume).

Answer: We rewrite the least squares minimization in matrix form,

$$\min (y - Xa)^T (y - Xa)$$

Expand,

$$\min y^T y - a^T X^T y - y^T X a + a^T X^T X a$$

Note that $a^T X^T y = y^T X a$ since they are scalars.

$$\min y^T y - 2a^T X^T y + a^T X^T X a$$

Now we take the gradient w.r.t. a and set it to 0 to minimizine,

$$\nabla_a y^T y - 2a^T X^T y + a^T X^T X a = -2X^T y + 2X^T X a = 0$$
$$\iff X^T X a = X^T y$$

We assume $X^T X$ is invertible,

$$a = (X^T X)^{-1} X^T y$$

Thus we can solve for a in polynomial time since it is just an inversion and matrix multiplication.

Exercise 4: (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 firehouse, your solution is allowed to use $O(k \log m)$ firehouse and have cost at most $(1 + \epsilon)$OPT.

Answer: Let $M = \{1, 2, \ldots, m\}$ index the set of all possible locations. Then, let x_{ij} represent the indicator of whether or not house c_i is closest to the firehouse located at position j. We write this firehouse problem as an LP,

$$\min \sum_{v_i \in V} \sum_{j \in M} d(v_i, \text{location } j) x_{ij}$$

$$\text{subject to } \sum_{j \in M} x_{ij} = 1 \quad \forall\ v_i \in V (\text{only one closest firehouse for each house})$$

$$\sum_{j \in M} y_j \leq k (\text{only } k \text{ firehouse})$$

$$x_{ij} \leq y_j \quad i \in V, j \in M$$

$$0 \leq x_{ij}, y_j \leq 1 \quad i \in V, j \in M$$

Where the last constraint is the integer programming constraint relaxed to LP. Note that the constraints with the y_i's are needed because multiple houses can have the same closest firehouse. So these constraints are the relaxation of the constraint $\max(x_{1i}, x_{2i}, \ldots, x_{ni}) = y_i, y_i = \{0, 1\}, \sum_{i=1}^{m} y_i = k$ which is the actual constraint on the firehouses. Thus, the y_i's can be thought of as if position j is a firehouse location if it equals 1.

The solution begins by solving this LP and getting the fractional solutions. Let $\hat{y}, \hat{x}$ be its solution. Now for the rounding algorithm. For any $\epsilon > 0$ and $\delta \in [0, 1]$, we select $(1 + 1/\epsilon)k \log(m/\delta)$ positions randomly, where position j has probability $\hat{y}_j/k$. For each position selected, set $y_j = 1$ and all the positions not selected, set them to 0. Note that the optimal positions will be a subset of the $\hat{j}_j$ that are not 0.

We wish to show that these randomly selected position cover the optimal positions. Let $V_i = \{j \in M : d(v_i, \text{location } j) \leq (1+\epsilon)\hat{C}_i\}$. Where $\hat{C}_i = \sum_{j\in M} d(v_i, \text{location } j)\hat{x}_{ij}$. Think of this as all the "smart" possible locations for firehouses for house i. Then, for each $i \in V$ and $\epsilon > 0$,

$$\sum_{j\in V_i} \hat{y}_j \geq \sum_{j\in V_i} \hat{x}_{ij} > \frac{\epsilon}{1+\epsilon}$$

Suppose otherwise that $\sum_{j\in V_i} \hat{x}_{ij} \leq \frac{\epsilon}{1+\epsilon}$. Then,

$$\begin{aligned}
\hat{C}_i &= \sum_{j\in V} d(v_i, \text{location } j)\hat{x}_{ij} \\
&\geq \sum_{j\notin V_i} d(v_i, \text{location } j)\hat{x}_{ij} \\
&> (1+\epsilon)\hat{C}_i \text{location } j)\hat{x}_{ij} \\
&\geq (1+\epsilon)\hat{C}_i \text{location } j) \left(1 - \frac{\epsilon}{1+\epsilon}\right) \text{ by assumption} \\
&= \hat{C}_i \text{ which is a contradiction.}
\end{aligned}$$

From this little proof, the probability that position i is covered by a randomly selected position is more than

$$\frac{\epsilon}{k(1+\epsilon)}$$

Since the we are selecting $(1+1/\epsilon)k\log(m/\delta)$ positions, the probability that position i is not covered is smaller than

$$\left(1 - \frac{\epsilon}{k(1+\epsilon)}\right)^{(1+\frac{1}{\epsilon})k\log\frac{m}{\delta}} < \frac{\delta}{m}$$

There are m houses, so the probability that one of them is not covered is at most $m\frac{\delta}{m} = \delta$. Thus, with high probability, we have that all the optimal positions are covered by our randomly selected set. The randomly selected set is of size $O(k\log m)$. Because we relaxed the integer program, we have that the cost is at most $(1+\epsilon)$OPT. **Note:** the paper "ϵ-Approximations with Minimum Packing Constraint Violation" by Lin and Vitter was used to generate this solution.

Exercise 5: In class we designed a 3/4-approximation for MAX-2SAT using LP rounding. Extend it to a 3/4-approximation for MAX-SAT (i.e., where clauses can have 1 or more variables). Hint: you may also need the following idea: if a clause has size k and we randomly assigne values to the variables (i.e., 0/1 with equal probability) then the probability we satisfy it is $1-1/2^k$.

Answer: In class, we discussed a rounding strategy. Assigning variable x_i in SAT problem with value 1 with probability x_i from the LP solution. Recall that this LP is,

$$\max \sum z_j$$
$$z_j \leq \sum_{\text{pos vars}} x_i \sum_{\text{neg vars}} (1-x_i)$$
$$0 \leq x_i, z_i \leq 1$$

Where z_j are the clausses and x_i are the variables. Let z_j have k variables. So we do the rounding strategy above of randomly assigning x_i to 1 with probability x_i. Then, the expected number of correct clauses is,

$$E[\text{clauses correct}] = 1 - \prod_{\text{pos vars}} (1-x_i) \prod_{\text{neg vars}} x_i$$

From the AM-GM inequalty,

$$\geq 1 - \left(\frac{\sum_{\text{pos vars}} (1-x_i) \sum_{\text{neg vars}} x_i}{k} \right)^k$$

From the constraints on z_j in the LP,

$$\geq 1 - \left(\frac{k - z_j}{k} \right)^k = 1 - \left(1 - \frac{z_j}{k}\right)^k$$

Consider the function $z_j \left(1 - \left(1 - \frac{1}{k}\right)^k\right)$, this equals the function above at $z_j = \{0,1\}$. Since the above function is concave on $[0,1]$, then we have

$$\geq z_j \left(1 - \left(1 - \frac{1}{k}\right)^k\right) = \alpha_k z_j$$

We now consider running this rounding scheme versus just randomly assigning all the clauses. With 1/2 probability, we'll run the rounding scheme and with 1/2 probability, we randomly assign all the variables. This means, the expected number of clauses that are correct will be greater than

$$z_j \left(\frac{1}{2}\alpha_k + \frac{1}{2}(1 - \frac{1}{2^k}) \right)$$

We now show the term in the bracket is greater than 3/4.

$$\frac{1}{2}\left(1 - \left(1 - \frac{1}{k}\right)^k\right) \frac{1}{2}(1 - \frac{1}{2^k}) \geq \frac{3}{4}$$
$$\left(1 - \frac{1}{k}\right)^k + \frac{1}{2^k} \leq \frac{1}{2}$$

This works for $k = 1, 2$. For $k \geq 3$, we have that $(1 - \frac{1}{k})^k \leq \frac{1}{e}$ and $\frac{1}{2^k} \leq \frac{1}{8}$. Thus,

$$\left(1 - \frac{1}{k}\right)^k + \frac{1}{2^k} \leq \frac{1}{2} \leq \frac{1}{e} + \frac{1}{8} \leq \frac{1}{2}$$

So it works for all k and so we have the expected number of clauses correct will be 3/4 approximate. **Note:** the following source was reference http://www.cs.tau.ac.il/∼azar/Methods-Class6.pdf.

Exercise 6: You are given data containing grades in different courses for 5 students. As discussed in Lecture 5, we are trying to "explain" the grades as a linear function of the student's aptitude, the easiness of the course and some error term. Denoting by Grade_{ij} the grade of student i in course j this linear model hypothesizes that

$$\text{Grade}_{ij} = \text{aptitude}_i + \text{easiness}_j + \epsilon_{ij}$$

where ϵ_{ij} is an error term.

As we saw in class, the problem of finding the best model that minimizes the sum of the $|\epsilon_{ij}|$'s can be solved by an LP. Your goal is to use any standard package for linear programming (Matlab/CVX, Freemat, Sci-Python, Excel, etc.; we reccommend CVX on matlab) to fit the best model to this data. Include a printout of your code, and the calculated easiness values of all the courses and the aptitudes of all the students.

	MAT	CHE	ANT	REL	POL	ECO	COS
Alex			C+	A	B+	A-	C+
Billy	B+	A-			A-	B	B
Chris	B	B+			A	A-	B+
David	A		B-	A		A-	
Elise		B-	C	B+	B	B	C

Assume $A = 10, B = 8$ and so on. Let $B+ = 9$ and $A- = 9.5$. (If you use a different numerical conversion please state it clearly.)

Answer: I used the following grade to point scale,

	A	A-	B+	B	B-	C+	C
Score	10	9.5	9	8	7.5	7	6

We wish to accomplish minimizing the error,

$$\min \sum_{i,j} |\epsilon_{ij}| \iff \min \sum_{i,j} |\text{Grade}_{ij} - \text{aptitude}_i - \text{easiness}_j|$$

This is nonlinear, so we linearize it,

$$\min \sum_{i,j} s_{ij}$$

$$\text{subject to } s_{ij} \geq 0$$

$$-s_{ij} \leq \text{Grade}_{ij} - \text{aptitude}_i - \text{easiness}_j \leq s_{ij}$$

Note that we have 25 grades, but this LP will have 75 variables. 25 for aptitude, 25 for easiness, and 25 slack variables. Below is the MATLAB code that I used to solve this LP with the results:

Student	Aptitude
Alex	-14.08
Billy	-13.52
Chris	-13.58
David	-13.58
Elise	-15.08

Course	Easiness
MAT	22.52
CHE	22.58
ANT	21.08
REL	24.08
POL	23.08
ECO	23.08
COS	21.40

```
f = [5;4;8;1;3;5;3;4;6;2;1;10;1;6;1;0;0;0;0;0];
A = -[eye(15), [
        1,0,0,-1,-1,0,0,0,0,1,0,0,0,0,0;
        0,1,0,0,0,-1,1,0,0,0,0,0,0,0,0;
        0,0,1,1,0,0,-1,-1,-1,0,0,0,0,0,0;
        0,0,0,0,1,0,0,1,0,-1,-1,-1,1,0,0;
        0,0,0,0,0,1,0,0,1,0,1,0,-1,-1,1;
]'];
b = [-1;-1;-1;0;0;0;0;0;0;0;0;0;0;0;0];

LB = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;  ;  ;  ;  ;];
UB = [];

y = linprog(f,A,b,[],[],LB,UB)

f'*y
5:02am
% first 25 are slack, second 5 are aptitude, third 7 easiness
f = [1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;
      1;1;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0];
A = [
    -1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0;
    0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0;
    0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0;
    0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0;
    0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1;
    0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0;
    0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0;
    0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0;
    0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0;
```

```
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
−1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,1 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,0 ,0 ,0 ,0 ,1;
−1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0;
0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0;
0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0;
0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0;
0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1;
0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0;
0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
    −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
    0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 , −1 ,0 ,0 ,0 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
    0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 , −1 ,0 ,0 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
    0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 , −1 ,0 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
    0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 , −1 ,0 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
    0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 , −1 ,0;
    0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,
    0 ,0 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 , −1 ,0 ,0 ,0 ,0 ,0 ,0 , −1;
];

% Use the grading table reading each row left to right before moving to
    the
% next row
b = [7;10;9;9.5;7;9;9.5;9.5;8;8;8;9;10;9.5;9;10;7.5;10;
    9.5;7.5;6;9;8;8;6;−7;−10;−9;−9.5;−7;−9;−9.5;−9.5;−8;
    −8;−8;−9;−10;−9.5;−9;−10;−7.5;−10;−9.5;−7.5;−6;−9;−8;−8;−6];

Aeq = [];
beq = [];
LB = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0; −Inf;−Inf;−Inf;−
    Inf;−Inf;−Inf;−Inf;−Inf;−Inf;−Inf;−Inf;−Inf];
UB = [];
```

Exercise 7: (Optimal life partners via MDP) Your friend is trying to find a life partner by going on dates with n people selected for her by an online dating service. After each date she has two choices: select the latest person she dated and stop the process, or reject this person and continue to date. She has asked you to suggest the optimum *stopping rule.* You can assume that the n persons are all linearly orderable (i.e. given a choice between any two, she is not indifferent and prefers one over the other). The dating service presents the n chosen people in random order, and her goal is to maximise the chance of ending up with the person that she will like the most among these n. (Thus ending up even with her second second favorite person out of the n counts as failure; she's a perfectionist.) Represent her actions as an MDP, compute the optimum strategy for her and the expected probability of success by following this strategy.

(Hint: The Optimal rule is of the form: *Date γn people and decide beforehand to pass on them. After that select the first person who is preferable to all people seen so far. You may also need that* $\sum_{k=t_1}^{t_2} \frac{1}{k} \approx \log \frac{t_2}{t_1}$)

Answer: Determine the state space of the MDP to be all the times t with 3 different states for each t. State 1 is that the t^{th} person is the best seen. State 0 is that the t^{th} person is not the best seen. State -1 is that you have already chosen a partner. Define the expected payoff for passing on the t^{th} person be $p(t)$.

For the optimal strategy, if $\frac{t}{n} > p(t)$, we'll choose the person at time t. This is from the fact that the probability of the best person being in the first t dates is,

$$= \frac{\binom{n-1}{t-1}}{\binom{n}{t}} = \frac{t}{n}$$

Thus if the payoff is less than the probability the the best person has already appeared, we should stop. We now define the payoof,

$$p(t-1) = \frac{t-1}{t} p(t) + \frac{1}{t} \max(\frac{t}{n}, p(t))$$

This equation comes from dynamic programming and can be explained as follows. There is a $\frac{t-1}{t}$ chance that the t^{th} person is not the best so far, which we skip for sure. The second term comes from the choice we have to make when the t^{th} person is the best. Either skip (get payoff $p(t)$) or accept them, in which case, they are the best with probability $\frac{t}{n}$. Now notice that $p(t)$ is a decreasing is decreasing in t since if $p(t)$ is the max, then $p(t-1) = p(t)$. Also, $\frac{t}{n}$ is increasing in t. Thus, we argue that these will cross and so this optimal strategy will work. We just need to calculate when they cross.

So the strategy is to date r people and then pick. The probability we stop at t and they and this is the best person is $\frac{1}{n} \cdot \frac{r}{t}$ from the fact that the best person overall and the best person in the first t dates is independent. So summing over all potential stopping times,

$$p(t-1) = \sum_{\tau=r}^{n} \frac{1}{n} \cdot \frac{r}{\tau} = \frac{r}{n} \sum_{\tau=r}^{n} \frac{1}{\tau}$$

Since the optimal stopping time is when $\frac{r}{n} > p(t-1)$, we have

$$\begin{gathered} \frac{r}{n} \sum_{\tau=r}^{n} \frac{1}{\tau} \leq \frac{r}{n} \\ \sum_{\tau=r}^{n} \frac{1}{\tau} \leq 1 \\ \implies \ln \frac{n}{r} \approx 1 \\ \implies r \approx \frac{n}{e} \end{gathered}$$

Thus, they should date roughly $\frac{1}{e} \approx 37\%$ of the people. This implies the rate of success is $\frac{r}{n} = \frac{1}{e} \approx 37\%$ which is pretty good for arbitrary large number of people.

Exercise 8: (extra credit) In question 3 try to design an algorithm that uses k firehouses but has cost $O(\text{OPT})$. (Needs a complicated dependent rounding; you can also try other ideas.) Partial credit available for progress.

Answer: This solution is from Lin and Vitter's paper "Approximation Algorithms for Geometric Median Problems". The solution is quite long so I will just give the main idea of the rounding procedure with high level commentary.

The solution starts the same way we did in question 3, solve the LP relaxation with fractional solutions $\hat{y}, \hat{x}$. Then compute for every i, $\hat{C}_i = \sum_{j \in M} d(v_i, j)\hat{x_i j}$. Define the neighbourhood

$$V_i = \{j \in M : d(v_i, \text{location } j) \leq (1+\epsilon)\hat{C}_i\}$$

and define the extended neighbourhood as $\bar{V}_i$ of house i iff one of the two rules hold,

1. $d(v_i, j) \leq (1+\epsilon)\hat{C}_j$
2. There exists a position j' such that $d(v_i, j') \leq (1+\epsilon)\hat{C}_j$ and $d(v_j, j') \leq (1+\epsilon)\hat{C}_j$, that is, $V_i \cap V_j = \emptyset$

So far, we can think of the extended neighbourhoods as being very reasonable guesses as to where the locations of the firehouses will be. Next, we sort the set $\{\bar{V}_i\}_{i \in V}$ by $\hat{C}_i$ in decreasing order. This sorts the neighbourhoods by the largest radii. Choose the set $\bar{V}_i$ with the smallest $\hat{C}_i$ and delete any set $\bar{V}_j$ such that $j \in \bar{V}_j$. This procedure formalizes the thought process of trying to remove overlapping of firehouses. Repeat this process until no extended neighbourhoods remain. We then pick all the extended neighbourhoods that weren't deleted and use this as our median set. Again, this process starts with small neighbourhoods (i.e., ones that the LP deem that the firehouses are close to) and removes any overlap. This most likely removes neighbourhoods with large radii. Large radii are bad because it basically means there is no good guess for the closest firehouse location. Iterating this procedure is like trying to cover the graph using all the highly likely areas first and then filling in the gaps. The paper goes on to prove that this only require $(1+\epsilon)k$ firehouses while being very close to the optimal cost.