Homework 3

Due: November 12, 11:59pm.

Email your PDF as an attachment to cmpe587homework@gmail.com

The attached file should be called `homework3-yourfirstname-yourlastname.pdf`

Homework policy: Please try to do the homework by yourself. If you get stuck, working in a group of two is okay, three at the most. Naturally, acknowledge any sources you worked with at the top of your solutions. LaTeX typesetting with pdf output is mandatory. Questions about the homework, LaTeX, or other course material can be asked on Piazza.

---

1. (For this problem you will need to know how the different chess (satranç) pieces move on a chess board. If you don't know, see `http://en.wikipedia.org/wiki/Chess#Movement` or `http://tr.wikipedia.org/wiki/Satranc#Hamleler`.)

   A rook (kale) starts out in the bottom-left corner of a chess board. It then takes a "random walk", where at each step it makes a random move. (I.e., it chooses uniformly at random from all legal moves it can make.) What is the expected value of the number of moves it makes before it returns again to the bottom-left corner of the board?

   Also solve this problem for two more chess pieces of your choice, from the following list: king (şah), bishop (fil), knight (at), queen (vezir).

2. Consider the path graph P_{n+1} on $n+1$ vertices, named $0, 1, 2, \ldots, n$, where there is an edge $(i, i+1)$ for each $0 \le i < n$. Recall that for $0 \le u, v \le n$, we write $M_{u,v}$ for the expected number of steps it takes for a random walk starting at vertex u to reach vertex v. Show that $M_{0,n} = n^2$. (Hint: try to prove a simple formula for $M_{i,i+1}$, for $i = 0, 1, 2, \ldots$.)

3. Alice wants to send a message to Bob, the message being a sequence of $d+1$ symbols from the field $\mathbb{F}_{256}$. Unfortunately, the channel she uses for transmission is noisy, and it might *corrupt* up to k symbols. (The corruptions are worst-case; it means that if Alice transmits the sequence $(a_1, \ldots, a_n)$ then Bob will receive some sequence $(b_1, \ldots, b_n)$, where the only thing you know is that $a_i = b_i$ for at least $n-k$ different i's.) Alice uses the following strategy to send the message $(m_0, m_1, \ldots, m_d)$. First, she imagines the polynomial $P(X) = m_d X^d + m_{d-1} X^{d-1} + \cdots + m_1 X + m_0$. Then, she transmits the "encoded" sequence $(P(0), P(1), \ldots, P(d+2k))$ along the channel. (Assume $d + 2k < 256$.)

   Prove that Bob can exactly determine Alice's intended message $(m_0, m_1, \ldots, m_d)$. Note: you do **not** have to show an *efficient* algorithm for Bob; just *some* algorithm.

4. (*Some tasks from Learning Theory.*) For the following problem you are given n "data points": each is a pair (X_i, y_i), where $X_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. (For intuition, you might like to think of the special case $d = 1$. In general, though, think of X_i as a column vector of height d.)

   (a) Suppose you wish to find the best-fitting hyperplane in "absolute error loss": this means you want to find the hyperplane $ax + b = 0$ (where $a \in \mathbb{R}^{1 \times d}$ is a row vector and $b \in \mathbb{R}$) which minimizes $\sum_{i=1}^{n} |aX_i + b - y_i|$. Formulate the task as a linear program.

(b) Suppose that each y_i is either $+1$ or -1. We say $ax + b = 0$ is a *weakly separating hyperplane* for the data points if $aX_i + b \geq 0$ whenever $y_i = +1$ and $aX_i + b \leq 0$ whenever $y_i = -1$. Formulate a linear program which is feasible (i.e., has a solution) if and only if there is a weakly separating hyperplane.

(c) Continuing the above, say that $ax + b = 0$ is a *separating hyperplane* if $aX_i + b > 0$ whenever $y_i = +1$ and $aX_i + b < 0$ whenever $y_i = -1$. Show that linear programming can be used to decide if there is a separating hyperplane.

(d) Continuing the above, suppose there is no separating hyperplane. Show that the hyperplane which minimizes "hinge loss" can be found using linear programming. Here the hinge loss of $ax+b$ on a point $(X_i, +1)$ is defined to be 0 if $aX_i + b \geq 1$ and $1 - (aX_i + b)$ otherwise; the hinge loss on a point $(X_i - 1)$ is defined to be 0 if $aX_i + b \leq -1$ and $(aX_i + b) - (-1)$ otherwise.

(e) For all of the above problems, what if instead of hyperplanes we are looking for "quadratic surfaces"? This means equations of the form

$$\sum_{1 \leq k \leq \ell \leq d} c_{k\ell} x_k x_\ell + \sum_{1 \leq k \leq d} a_k x_k + b = 0.$$

Show that we can still solve all of the problems using polynomial-sized linear programs.

5. Recall the Min-st-Cut problem. Given is a directed graph $G = (V, E)$, a nonnegative "cost" c_{uv} for each edge $uv \in E$, and two special vertices: the "source" vertex $s \in V$ and the "target" vertex $t \in V$. The task is to find an st-cut $S \subseteq V$ of minimal cost. Here an "st-cut" S means that $s \in S$, $t \notin S$, and its cost is

$$\text{cost}(S) = \sum_{\substack{uv \in E \\ u \in S, v \notin E}} c_{uv}.$$

Consider the following Integer Linear Program, which has a variable x_{uv} for each edge $uv \in E$:

$$\begin{aligned} \text{minimize} \quad & \sum_{uv \in E} c_{uv} x_{uv} && && (1) \\ \text{such that} \quad & x_{uv} \in \{0, 1\} && \text{for all } uv \in E && (2) \\ & \sum_{\text{edges } (u,v) \text{ along } P} x_{uv} \geq 1 && \text{for all paths } P \text{ from } s \text{ to } t. && (3) \end{aligned}$$

(a) Explain why this ILP *exactly* captures the Min-st-Cut problem.

Let's consider the LP you get when you "relax" the constraint (2) to $0 \leq x_{uv} \leq 1$. Although this is an LP, it's not clear that you can solve it in polynomial time, because it actually might have exponentially many constraints! The reason is that there might be exponentially many paths in the graph from s to t.

However, it is a wonderful (though difficult-to-prove) fact about linear programming that you can efficiently solve such LPs as long as you have an efficient "separation oracle". What this means is the following: You need to show that there is an algorithm that, when given candidate values for the variables x_{uv}, either outputs "Yes, these satisfy all the constraints", or outputs, "No, here is a specific constraint that is not satisfied."

(b) Since it's easy to check whether all the given x_{uv}'s are between 0 and 1, having an efficient "separation oracle" amounts to having an efficient algorithm that decides whether (3) is satisfied, and if not, outputs a specific unsatisfied constraint. Explain briefly why this is possible. (Hint: you might like to think of "x_{uv}" as representing the "length" from u to v. You are allowed to appeal to some well-known basic algorithmic theory.)

So we can actually solve the LP in polynomial time. Still, this doesn't necessarily solve the Min-st-Cut problem, because the solution to the LP can be "fractional"; i.e., the optimal x_{uv}'s don't have to be in $\{0, 1\}$. However, we will now describe a good randomized "rounding algorithm".

Suppose the algorithm obtains an optimal LP solution $(x^*_{uv})_{uv \in E}$. As mentioned in the hint above, the algorithm will think of x^*_{uv} as representing the "length" of the edge from u to v. Next, suppose the algorithm picks a uniformly random real number $0 \leq \boldsymbol{\rho} < 1$. Finally, the algorithm defines $S = \{v \in V : \text{"minimum-distance}(s, v)\text{"} \leq \boldsymbol{\rho}\}$.

(c) Show that the resulting S is a valid st-cut.

(d) Show that in expectation, $\text{cost}(S) \leq \text{LPOpt} := \sum_{uv \in E} c_{uv} x^*_{uv}$. (Hint: the main step is to show $\mathbf{Pr}[u \in S, v \notin S] \leq x^*_{uv}$.)