CS 174: Combinatorics and Discrete Probability Fall 2012

Homework 11

This homework is not for submissiom

In this homework, we will see a few of the numerous applications of the weighted majority algorithm. It is remarkable how powerful a simple algorithm, such as the multiplicative update rule, can be. The problem statements here may appear long, but that is mainly because this homework is combining some aspects of lecture notes and practice problems. It is suggested that you attempt these problems in preparation for the final exam. The solutions to these problems will be posted in the middle of next week.

Problem 1. (*von Neumann's Minmax Theorem*). The first problem is one in game theory. We consider a game between two players, where player 1 can choose from among n options and player 2 can choose from among m options. The payoff to player 1 is given by a payoff matrix, $A_{n\times m}$, where A_{ij} is the payoff received by player 1 if she plays i and player 2 plays j. We will consider *zero-sum* games, thus, the payoff matrix from the second player will simply be $-A$. We will assume that each entry of the matrix A is bounded in the range $[-M, M]$.

The players may play according to *mixed strategies*, *i.e.* instead of choosing a single option, they choose a distribution over the available options. Suppose, player 1 chooses a distribution, $\mathbf{x} \in \Delta_n$, and player 2 chooses a distribution, $\mathbf{y} \in \Delta_n$, then the expected payoff received by player 1 is $\mathbf{x}^T A\mathbf{y}$.

In order to consider the value of the game, it is useful to think of one player going first and announcing her strategy, and the second player using *best response*. Suppose player 1 plays first, then she guarantee that her expected payoff is $v_1 = \max_{\mathbf{x}\in\Delta_n} \min_{\mathbf{y}\in\Delta_m} \mathbf{x}^T A\mathbf{y}$. This is the *max-min* value, denoted by v_1. Here, the second player tries to minimize the expected payoff of player 1 (since it is a zero-sum game). One way to think of this as follows, suppose player 1 is going to use $\mathbf{x}$ as her strategy and she announces it. In that case, player 2 will choose $\mathbf{y}$ that minimizes, $\mathbf{x}^T A\mathbf{y}$. Player 1 should choose the vector $\mathbf{x}$ that maximizes this *minimum value*.

Similarly, if player 2 goes first, and announces $\mathbf{y}$. Then the maximum expected value that player 1 can obtain is $\min_{\mathbf{y}\in\Delta_m} \max_{\mathbf{x}\in\Delta_n} \mathbf{x}^T A\mathbf{y}$. This is the *min-max* value, denoted by $\bar{v}_1$. The min-max theorem, states that $v_1 = \bar{v}_1$, or that the max-min and min-max values are the same. Thus, who plays first is irrelevant.

We will prove the min-max theorem using the repeated n-decision problem. Suppose, the two players play the same game for time steps $t = 1, \ldots, T$. Player 1 goes first and plays using the weighted majority algorithm. Player 2 plays best response, *i.e.* if player 1 plays $\mathbf{x}^t \in \Delta_n$ at time step t, player 2 chooses $\mathbf{y}^t$, that minimizes $(\mathbf{x}^t)^T A\mathbf{y}^t$.

1. First show that the *max-min* value is smaller than the *min-max* value, *i.e.* $v_1 \le \bar{v}_1$. (This should obviously be the case, since going second is an advantage.)
   Solution: $\max_{\mathbf{x}\in\Delta_n} \mathbf{x}^T A\mathbf{y} \ge \mathbf{x}^T A\mathbf{y}$ for every $\mathbf{x}$ and $\mathbf{y}$. Now taking the min with respect to $\mathbf{y}$, we have $\bar{v}_1 = \min_{\mathbf{y}\in\Delta_m} \max_{\mathbf{x}\in\Delta_n} \mathbf{x}^T A\mathbf{y} \ge \min_{\mathbf{y}\in\Delta_m} \mathbf{x}^T A\mathbf{y}$ for every $\mathbf{x}$. Therefore the inequality remains even when taking max over $\mathbf{x}$ of the RHS. Thus, $\bar{v}_1 \ge v_1$.

2. Show that the average payoff of player 1, $\frac{1}{T}\sum_{t=1}^{T}(\mathbf{x}^t)^T A\mathbf{y}^t$ is at most v_1.
   Solution: We consider the repeated n-decision problem described above. Since player 1 goes first on each round, she can never achieve more than $v_1 = \max_{\mathbf{x}\in\Delta_n} \min_{\mathbf{y}\in\Delta_m} \mathbf{x}^T A\mathbf{y}$, on any given round. Thus, the average payoff of player 1 is at most v_1.

3. Show that the average payoff that player 1 could have obtained in hindsight is at least $\bar{v}_1$.
   Solution: Suppose $\mathbf{y}^1, \ldots, \mathbf{y}^T$ are the actual vectors played by player 2 in the game. Let $\bar{\mathbf{y}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{y}^t$. The best average payoff in hindsight is $\max_{\mathbf{x}\in\Delta^n} \mathbf{x}^T A\bar{\mathbf{y}}$. Clearly this quantity is at least $\bar{v}_1$.

4. Use the weighted majority theorem in the limit as $T \to \infty$ to conclude the fact that $v_1 = \bar{v}_1$.
   Solution: The Weighted Majority Theorem guarantees that the average regret is at most $O(M\sqrt{\frac{\log(n)}{T}})$. Let M be a bound on the absolute value of the all entries in A (this can always be done since A is finite). Then, this implies $\bar{v}_1 - v_1 \leq O(M\sqrt{\frac{\log(n)}{T}})$ for every value of T. Taking limit as $T \to \infty$ implies that $v_1 = \bar{v}_1$.

Problem 2. (*Linear Programming*) A typical linear program has the following form:

$$\min \ \mathbf{c}^T \cdot \mathbf{x}$$
$$\text{subject to:}$$
$$A\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \geq 0$$

where $\mathbf{x} \in \mathbb{R}^m$, $x_i \geq 0$ for $i = 1, \ldots, n$. A is an $n \times m$ matrix, $\mathbf{b} \in \mathbb{R}^n$. Thus, there are n constraints; and $\mathbf{c} \in \mathbb{R}^m$. Let $\mathbf{a}_i$ denote the i^{th} row of A and b_i denote the i^{th} entry of the vector $\mathbf{b}$. Thus, $\mathbf{a}_i^T\mathbf{x} \leq b_i$ is simply the i^{th} constraint of a linear program. One way to solve a linear program is to guess the value of the objective function, say z^* (by binary search) and try to find a vector in the set $\mathcal{P} = \{\mathbf{x} \mid \mathbf{x} \geq 0, \mathbf{c}^T\mathbf{x} = z^*\}$. The superscript, ^{T}, here means the transpose.

1. Let $\mathbf{a} \in \mathbb{R}^m$ and $b \in \mathbb{R}$. Show that if there is only one constraint, $\mathbf{a}^T\mathbf{x} \leq b$, then it is easy to determine if there exists $\mathbf{x} \in \mathcal{P}$ for which the constraint is satisfied.
   Solution: The observation here is that this is a linear program with only one constraint. While still simple, it is not trivial. So you may ignore this part of the problem. We may assume without loss of generality that $z^* > 0$. Then the program is feasible if and only if $\min_{i\in[m],c_i>0} \frac{a_i z^*}{c_i} \leq b$.

2. Now, let $\rho = \max\{\max_{i,\mathbf{x}\in\mathcal{P}} |\mathbf{a}_i^T\mathbf{x} - b_i|, 1\}$. We will set up a repeated n-decisions problem. Each constraint is thought of as one of n choices. Suppose $\mathbf{w}^t$ is the distribution over the n constraints obtained by playing weighted majority at time step t. Let $\mathbf{a}^t = (\mathbf{w}^t)^T A$ and let $b^t = (\mathbf{w}^t)^T\mathbf{b}$. We consider the constraint $(\mathbf{a}^t)^T\mathbf{x} \leq b^t$. Show that if there is no $\mathbf{x} \in \mathcal{P}$ that satisfies $(\mathbf{a}^t)^T\mathbf{x} \leq b_i$, then the original program is *infeasible*, *i.e.* there is no $\mathbf{x} \in \mathcal{P}$, such that $A\mathbf{x} \leq \mathbf{b}$.
   Solution: Suppose the original program was feasible, *i.e.* $\exists\mathbf{x} \in \mathcal{P}$, such that $A\mathbf{x} \leq \mathbf{b}$. Then for such an $\mathbf{x}$, for any $\mathbf{w} \in \Delta_n$, $\mathbf{w}^T A\mathbf{x} \leq \mathbf{w}^T\mathbf{b}$. Thus, if the constraint $(\mathbf{a}^t)^T\mathbf{x} \leq b^t$ is infeasible, then the original program must be infeasible.

3. Otherwise, let $\mathbf{x}^t$ be any vector in the set $\mathcal{P}$ that satisfies the constraint, $(\mathbf{a}^t)^T\mathbf{x}^t \leq b_i$. The payoff for action (constraint) i is $(\mathbf{a}_i)^T \cdot \mathbf{x}^t - b_i$. Show, that the payoff of the algorithm is *negative* at each round, unless the program was declared *infeasible*.
   Solution: Note that the payoff of the algorithm is simply $\sum_{i=1}^n \mathbf{w}_i^t((\mathbf{a}_i)^T\mathbf{x}^t - b_i) = (\mathbf{w}^t)^T A\mathbf{x}^t - \mathbf{w}^t\mathbf{b}$ which is negative, since $\mathbf{x}^t$ was chosen to be a feasible point.

4. Let $\mathbf{x}^* = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}^t$ and suppose $T = 16\rho^2\ln(n)/\epsilon^2$. Then show that, $\mathbf{a}_i^t\mathbf{x}^* \leq b_i + \epsilon$ for each i.

Solution: Let $\text{payoff}_{\mathsf{WM}}$ be the total payoff obtained by playing the weighted majority algorithm. Note that this payoff is negative. The weighted majority algorithm guaratees that for each i,

$$((\mathbf{a}_i)^T \mathbf{x}^* - b_i) - \text{payoff}_{\mathsf{WM}} \leq 4\rho\sqrt{\frac{\ln(n)}{T}}$$

Since, $\text{payoff}_{\mathsf{WM}} < 0$ and substituting the value of T, we get $(\mathbf{a}_i)^T \mathbf{x}^* \leq b_i + \epsilon$ for every i.

Remark: What we have shown is that we have *almost* solved the linear program. Note that each constraint may be violated slightly (by ϵ), and a tighter guarantee may be obtained using a larger T. This is not the most optimal method to solve linear programs. More involved algorithms, such as the ellipsoid algorithm, do give *truly* polynomial time algorithms for linear programming. None the less, the above approach can be used to obtain interesting polynomial time approximation algorithms, when it is sufficient to find an *approximately* feasible solution to a linear program.

Problem 3. (*Sleeping Experts*). A variation of the standard n-decision problem, is the *sleeping experts problem.* Here, each of the n decisions is just some expert advice. However, at any given time step t, the decision-maker (your algorithm) may only have access to a subset, $S^t \subseteq [n]$ of experts. The remaining experts may be *sleeping.*

The notion of *regret* in hindsight is more involved in this case. Note that the payoff of the best expert makes little sense, because some experts may not be available (awake) on every round. We consider the following to be the best *strategy* in hind-sight: we consider a ranking over the experts. A ranking, σ is simply a permutation of n elements. At time-step t, the ranking strategy according to σ, is implemented as follows: let $\sigma(S^t)$ denote the *highest-ranked* expert that is in S^t according to ranking σ. For example, if $n = 5$, $\sigma = (3, 2, 4, 1, 5)$ and $S^t = \{1, 4\}$, then the strategy would be to follow the advice of expert 4, since the 4^{th} expert is awake and ranked higher than other awake experts (in this case just expert 1). Thus the regret is measured with respect to the best *ranking* in hindsight (when the payoffs for all experts are known). We assume that the payoffs for each expert lie in the range $[-M, M]$. Then,

$$\text{regret} = \max_{\sigma \in \text{perm}(n)} \frac{1}{T} \sum_{t=1}^{T} p^t_{\sigma(S^t)} - \frac{1}{T} \sum_{t=1}^{T} p^t_{d^t},$$

where d^t is the expert (from the set S^t), chosen by the decision-maker, and $\sigma(S^t)$ is the expert that would have been chosen by the ranking σ.

Show that it is possible to obtain an algorithm that after T time-steps has regret $O(M\sqrt{\frac{n\log(n)}{T}})$. Show that in fact you can assume that each of the $n!$ possible rankings is a new *meta-expert.* Now implement weighted-majority algorithm as if you were playing a repeated $n!$-decision game with $n!$ experts. You may assume that at the end of the round, the *entire* payoff vector is revealed, including the payoffs that experts who were sleeping (and were not available to the decision-maker) would have received. What can you say about the running time of your algorithm?

Solution: As indicated in the problem statement, let each permutation over the experts be a meta-expert. We use the weighted majority algorithm with $n!$ possible choices at each round. Note that whenever we receive the payoff information at the end of each round, it is possible to deduce what payoff each of the $n!$ *meta-experts* would have received. Thus, we can make the appropriate updates to weighted majority.

We use the fact that $\log(n!) = O(n \log(n))$ to get the required result. The running time at each time-step is $n!$ which is prohibitively expensive. However, it is unlikely that there is any polynomial time strategy that gives a non-trivial regret guarantee!