Exercise Set V, Advanced Algorithms 2022

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. Solve as many problems as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

These problems are taken from various sources at EPFL and on the Internet, too numerous to cite individually.

1 In this exercise we consider the Hedge algorithm and use the same notation as in the lecture notes. The average [external] regret of Hedge is defined as

$$\frac{\sum_{t\leq T}\vec{p}^{(t)}\cdot\vec{m}^{(t)}-\min_i\sum_{t\leq T}m_i^{(t)}}{T}$$

i.e., how much we "regret", on average over the days, compared to the best single strategy i.

1a If you knew the number of days T in advance, how would you set the parameter ϵ of Hedge to minimize the average external regret?

1b *(*)* Even if you do not know T in advance, describe a strategy that achieves roughly the same average external regret as in the case when T is known.

Hint: There is no need to redo the analysis from scratch. For example, you could consider restarting the algorithm each time you get to a day t of the form 4^i.

Solution:

1a By Theorem 16.2 from the lecture notes, the average external regret is at most

$$R(\epsilon):=\frac{\ln N}{\epsilon T}+\epsilon.$$

Minimizing this over ϵ, we get

$$\epsilon=\sqrt{\frac{\ln N}{T}}\quad\text{and}\quad R(\epsilon)=2\sqrt{\frac{\ln N}{T}}.$$

1b We proceed as in the hint. More precisely, we divide the time period $[1,T]$ into phases where the i-th phase has length 4^i (except the last phase, which could be shorter) – that is, into phases $[1,4]$, $[5,20]$, $[21,84]$, etc. We restart the algorithm at the beginning of each such phase, by resetting all weights to 1 and setting $\epsilon=\epsilon_i=\frac{\sqrt{\ln N}}{2^i}$.

Now let us analyze the regret. (We assume for simplicity that the last, k-th phase also has full length 4^k.) For each phase i we have that the average regret is at most $2\frac{\sqrt{\ln N}}{2^i}$. Thus the total regret in phase i is at most $2\cdot 2^i\cdot\sqrt{\ln N}$. The average regret over all phases (i.e. over the time $T=\sum_{i=1}^k 4^i\approx 4^{k+1}/3$) is thus at most

$$\frac{\sum_{i=1}^k 2\cdot 2^i\cdot\sqrt{\ln N}}{T}\approx\frac{2\cdot 2^{k+1}\cdot\sqrt{\ln N}}{T}\approx\frac{2\cdot\sqrt{3}\cdot\sqrt{T}\cdot\sqrt{\ln N}}{T}=2\sqrt{3}\cdot\sqrt{\frac{\ln N}{T}}.$$

2 Suppose you are using the Hedge algorithm to invest your money (in a good way) into N different investments. Every day you see how well your investments go: for $i \in [N]$ you observe the change of each investment in percentages. For example, change(i) $= 20\%$ would mean that investment i increased in value by 20% and change(i) $= -10\%$ would mean that investment i decreased in value by 10%.

How would you implement the "adversary" at each day t so as to make sure that Hedge gives you (over time) almost as a good investment as the best one? In other words, how would you set the cost vector $\vec{m}^{(t)}$ each day?

Solution: Recall that, in the Hedge algorithm we learned in class, the total loss over time is upper bounded by $\sum_{t=1}^{T} m_i^t + \frac{\ln N}{\epsilon} + \epsilon T$.

In the case of investments, we want to do almost as good as the best investment. Let g_i^t be the fractional change of the value of i'th investment at time t. I.e., $g_i^t = (100 + \mathit{change}(i))/100$, and $p_i^{t+1} = p_i^t \cdot g_i^t$. Thus, after time T, $p_i^{T+1} = p_i^1 \prod_{t=1}^{T} g_i^t$. To get an analogous bound to that of the Hedge algorithm, we take the logarithm. The logarithm of the total gain would be $\sum_{t=1}^{T} \ln g_i^t$. To convert this into a loss, we multiply this by -1, which gives a loss of $\sum_{t=1}^{T}(-\ln g_i^t)$. Hence, to do almost as good as the best investment, we make our cost vectors to be $m_i^t = -\ln g_i^t$.

Now, from the analysis of Hedge algorithm in the lecture, it follows that for all $i \in [N]$,

$$\sum_{t=1}^{T} p_i^{(t)} \cdot m^{(t)} \leq \sum_{t=1}^{T} m_i^{(t)} + \frac{\ln N}{\epsilon} + \epsilon T.$$

Taking the exponent in both sides, We have that

$$\begin{aligned}
\exp\left(\sum_{t=1}^{T} p_i^{(t)} \cdot m^{(t)}\right) &\leq \exp\left(\sum_{t=1}^{T} m_i^{(t)} + \frac{\ln N}{\epsilon} + \epsilon T\right) \\
\prod_{t=1}^{T} \exp(p_i^{(t)} \cdot m^{(t)}) &\leq \exp(\ln N/\epsilon + \epsilon T) \prod_{t=1}^{T} \exp(m_i^t) \\
\prod_{t=1}^{T} \prod_{i\in[N]} (1/g_i^t)^{p_i^{(t)}} &\leq \exp(\ln N/\epsilon + \epsilon T) \prod_{t=1}^{T} (1/g_i^{(t)})
\end{aligned}$$

Taking the T-th root on both sides,

$$\left(\prod_{t=1}^{T} \prod_{i\in[N]} (1/g_i^t)^{p_i^{(t)}}\right)^{(1/T)} \leq \exp(\ln N/\epsilon T + \epsilon) \left(\prod_{t=1}^{T} (1/g_i^{(t)})\right)^{(1/T)}.$$

This can be interpreted as the weighted geometric mean of the loss is not much worse than the loss of the best performing investment.

3 Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Consider the following linear program with n variables:

$$\begin{array}{ll}
\textbf{maximize} & c^T x \\
\textbf{subject to} & Ax = b \\
 & x \geq 0
\end{array}$$

Show that any extreme point x^* has at most m non-zero entries, i.e., $|\{i : x_i^* > 0\}| \leq m$.

Hint: what happens if the columns corresponding to non-zero entries in x^* are linearly dependent?

(If you are in a good mood you can prove the following stronger statement: x^* is an extreme point if and only if the columns of A corresponding to non-zero entries of x^* are linearly independent.)

Solution: Without loss of generality, index x^* so that $x^*_1, \ldots, x^*_k > 0$ and $x^*_{k+1}, \ldots, x^*_n = 0$. Also let $a_1, a_2, \ldots, a_k$ denote the columns of A corresponding to the nonzero variables $x^*_1, \ldots, x^*_k$.

We start by showing that if $a_1, \ldots, a_k \in \mathbb{R}^m$ are linearly dependent then x^* is not an extreme point. This implies that "any extreme point x^* has at most m non-zero entries, i.e., $|\{i : x^*_i > 0\}| \leq m$" since no more than m vectors can be linearly independent in the m-dimensional space $\mathbb{R}^m$.

Since we assume $a_1, \ldots, a_k$ to be linearly dependent we can write $\sum_{i=1}^k \lambda_i a_i = 0$ for some scalars λ_i that are not all equal to 0. Now it is easy to verify that for a small enough $\varepsilon > 0$,

$$y := \begin{bmatrix} x^*_1 \\ \vdots \\ x^*_k \\ x^*_{k+1} \\ \vdots \\ x^*_n \end{bmatrix} + \varepsilon \cdot \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_k \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{and} \quad z := \begin{bmatrix} x^*_1 \\ \vdots \\ x^*_k \\ x^*_{k+1} \\ \vdots \\ x^*_n \end{bmatrix} - \varepsilon \cdot \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

are both feasible solutions to the linear program and $x^* = \frac{y+z}{2}$ and thus x^* is not an extreme point.

We now complete the proof of the good mood problem by showing that if x^* is not an extreme point, then $a_1, \ldots, a_k$ are linearly dependent. If x^* is not an extreme point, then there is a vector $y \in \mathbb{R}^n \setminus \{0\}$ such that $x^* + y$ is feasible and $x^* - y$ is feasible. By simple rewriting, $Ax = b$ and $A(x + y) = b$ imply $Ay = 0$, i.e., $\sum_{i=1}^n a_i y_i = 0$. By the nonnegativity constraints, we must have $y_i = 0$ for all $i > k$ (for suppose that $y_i > 0$ and $x^*_i = 0$; then either $x^* - y$ or $x^* + y$ would need to be negative on coordinate i and thus infeasible). Hence we have that $\sum_{i=1}^k a_i y_i = 0$, which shows that the vectors $a_1, \ldots, a_k$ are linearly dependent.

In the proofs above we used that x^* is not an extreme point (i.e., it can be written as a convex combination of other feasible vectors) if and only if it can be written as a convex combination of **two** other feasible vectors. The proof here is as follows: suppose that $x^* = \sum_{i=1}^n \lambda_i y_i$ for $\lambda_1, ..., \lambda_n > 0$, $\sum_i \lambda_i = 1$ and $y_1 \neq x^*$. Then we can rewrite

$$x^* = \sum_{i=1}^n \lambda_i y_i = \lambda_1 y_1 + \sum_{i=2}^n \lambda_i y_i = \lambda_1 y_i + (1 - \lambda_1) \cdot \left(\sum_{i=2}^n \frac{\lambda_i}{1 - \lambda_1} y_i \right)$$

which is a convex combination of two other feasible points. The point $\sum_{i=2}^n \frac{\lambda_i}{1-\lambda_1} y_i$ is feasible as a convex combination of feasible points (note that $\sum_{i=2}^n \lambda_i = 1 - \lambda_1$).

4 Consider the following quadratic programming relaxation of the Max Cut problem on $G = (V, E)$:

$$\begin{array}{ll} \textbf{maximize} & \displaystyle\sum_{\{i,j\} \in E} (1 - x_i) x_j + x_i (1 - x_j) \\ \textbf{subject to} & x_i \in [0, 1] \ \ \forall i \in V \end{array}$$

Show that the optimal value of the quadratic relaxation actually equals the value of an optimal cut. (Unfortunately, this does not give an exact algorithm for Max Cut as the above quadratic program is NP-hard to solve (so is Max Cut).)

Hint: analyze basic randomized rounding.

Solution: We give a rounding algorithm that gives a cut x^* (an integral vector) of expected value equal to the value of an optimal solution x to the quadratic program. The rounding algorithm is the basic one: set $x_i^* = 1$ with probability x_i and $x_i^* = 0$ with probability $1 - x_i$. The expected value of the objective function is then $\sum_{i,j} \mathbb{E}\left((1 - x_i^*)x_j^* + x_i^*(1 - x_j^*)\right) = \sum_{(i,j)\in E}(1 - x_i)x_j + x_i(1 - x_j)$ by independence of the variables x_i^* and x_j^*. Thus the expected value after rounding is the optimal value of the quadratic relaxation. But as no integral cut can have larger value than the relaxation, the value of the obtained cut x^* must always equal the value of the fractional solution x. It follows that the optimal value of the quadratic relaxation equals the value of an optimal cut.