CS261: Exercise Set #6

For the week of February 8–12, 2016

Instructions:

(1) *Do not turn anything in.*

(2) The course staff is happy to discuss the solutions of these exercises with you in office hours or on Piazza.

(3) While these exercises are certainly not trivial, you should be able to complete them on your own (perhaps after consulting with the course staff or a friend for hints).

Exercise 26

In the online-decision making problem (Lecture #11), suppose that you know in advance an upper bound Q on the sum of squared rewards ($\sum_{t=1}^{T}(r^t(a))^2$) for every action $a \in A$. Explain how to modify the multiplicative weights algorithm and analysis to obtain a regret bound of $O(\sqrt{Q \log n} + \log n)$.

Exercise 27

Consider the thought experiment sketched at the end of Lecture #11: for a zero-sum game specified by the $n \times n$ matrix $\mathbf{A}$:

- At each time step $t = 1, 2, \ldots, T = \frac{4 \ln n}{\epsilon^2}$:
  - The row and column players each choose a mixed strategy ($\mathbf{p}^t$ and $\mathbf{q}^t$, respectively) using their own copies of the multiplicative weights algorithm (with the action set equal to the rows or columns, as appropriate).
  - The row player feeds the reward vector $\mathbf{r}^t = \mathbf{A}\mathbf{q}^t$ into (its copy of) the multiplicative weights algorithm. (This is just the expected payoff of each row, given that the column player chose the mixed strategy $\mathbf{q}^t$.)
  - The column player feeds the reward vector $\mathbf{r}^t = -(\mathbf{p}^t)^T\mathbf{A}$ into the multiplicative weights algorithm.

Let

$$v = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{p}^t)^T\mathbf{A}\mathbf{q}^t$$

denote the time-averaged payoff of the row player. Use the multiplicative weights guarantee for the row and column players to prove that

$$v \geq \left(\max_{\mathbf{p}} \mathbf{p}^T\mathbf{A}\hat{\mathbf{q}}\right) - \epsilon$$

and

$$v \leq \left(\min_{\mathbf{q}} \hat{\mathbf{p}}^T\mathbf{A}\mathbf{q}\right) + \epsilon,$$

respectively, where $\hat{\mathbf{p}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{p}^t$ and $\hat{\mathbf{q}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{q}^t$ denote the time-averaged row and column strategies.

[Hint: first consider the maximum and minimum over all deterministic row and column strategies, respectively, rather than over all mixed strategies $\mathbf{p}$ and $\mathbf{q}$.]

Exercise 28

Use the previous exercise to prove the minimax theorem:

$$\max_{\mathbf{p}} \left(\min_{\mathbf{q}} \mathbf{p}^T \mathbf{A} \mathbf{q} \right) = \min_{\mathbf{q}} \left(\max_{\mathbf{p}} \mathbf{p}^T \mathbf{A} \mathbf{q} \right)$$

for every zero-sum game $\mathbf{A}$.

Exercise 29

There are also other notions of regret. One useful one is *swap regret*, which for an action sequence $a^1, \ldots, a^T$ and a reward vector sequence $r^1, \ldots, r^T$ is defined as

$$\max_{\delta: A \to A} \sum_{t=1}^{T} r^t(\delta(a^t)) - \sum_{t=1}^{T} r^t(a^t)$$

where the maximum ranges over all functions from A to itself. Thus the swap regret measures how much better you could do in hindsight by, for each action a, switching your action from a to some other action (on the days where you previously chose a). Prove that, even with just 3 actions, the swap regret of an action sequence can be arbitrarily larger (as $T \to \infty$) than the standard regret (as defined in Lecture #11).[1]

Exercise 30

At the end of Lecture #12 we showed how to use the multiplicative weights algorithm (as a black box) to obtain a $(1-\epsilon)$-approximate maximum flow in $O(\frac{OPT^2}{\epsilon^2} \log n)$ iterations in networks where all edges have capacity 1. (We are ignoring the outer loop that does binary search on the value of OPT.) Extend this idea to obtain the same result for maximum flow instances in which every edge capacity is at least 1.

[Hint: if $\{\ell_e^*\}_{e \in E}$ is an optimal dual solution, with value $OPT = \sum_{e \in E} c_e \ell_e^*$, then obtain a distribution by scaling each $c_e \ell_e^*$ down by OPT. What are the relevant edge lengths after this scaling?]

[1] Despite this, there are algorithms (a bit more complicated than multiplicative weights, but still reasonably simple) that guarantee swap regret sublinear in T.