15-451/651 Algorithm Design & Analysis
Spring 2023, Recitation #11

Objectives

- Understand the objective of an online algorithm, how to create one, and how to bound its competitive ratio.
- Understand the analysis of the multiplicative weights algorithm by trying the analysis with different weight update rules.

Recitation Problems

1. **(Online Algorithms: Splay-Coin)** 15-451 is creating a new cryptocurrency: the splay-coin. To garner interest in the coin, we are allowing each coin to be exchanged for 1 bonus point to your final grade.

   Naturally, you would like to acquire 100 splay-coins by the last day of the n day semester, since you did not turn in any homework. On day i you check the price p_i of splay-coins, and decide whether to purchase. Unfortunately, due to a bug in the system, you can only purchase once, and want to minimize the cost when you do so.

   Since cryptocurrency values are often highly volatile, 15-451 has also implemented some price controls on our coin, including a lower bound L and upper bound U on the price p_i at any day i.

   (a) Being the clever 15-451 student that you are, you'd like to use your algorithms knowledge to help you minimize the price you pay. Come up with a $\sqrt{U/L}$-competitive algorithm for this problem.

   (b) Now just to make sure that none of your classmates can best you, prove that no deterministic algorithm can achieve a competitive ratio better than $\sqrt{U/L}$.

2. **(Expert analysis)** In lecture we saw that the simple procedure that multiplied the weight of each expert by $\frac{1}{2}$ whenever the expert made a mistake, resulted in

$$m = \text{ \#mistakes of algorithm } \leq 2.41(M + \log_2 n),$$

where M = #mistakes made by the best expert and n = # of experts. If we multiply the weight by 2/3 at each time, how does this analysis change? Let's see

- The total weight of the experts starts at _____
- Each time we make a mistake, the new total weight is at most _____ times the old weight
- If we make m mistakes and the best expert makes M mistakes, then

  __________ $\leq$ final total weight of all experts $\leq$ __________

- Therefore, $m \leq$ _________________________

3. **(Expert Minimax)** Recall Von Neumann's Minimax Theorem from game theory:

$$\mathsf{lb} = \max_{\mathbf{p}} \min_{\mathbf{q}} V_R(\mathbf{p}, \mathbf{q}) = \min_{\mathbf{q}} \max_{\mathbf{p}} V_R(\mathbf{p}, \mathbf{q}) = \mathsf{ub}$$

which effectively says that as long as both players are playing optimally, it doesn't matter which player declares their strategy first: the other player cannot gain an advantage by reacting and adjusting their strategy.

It turns out that one convenient way to prove this is using the multiplicative weights algorithm, which we will do here. The first direction of the inequality is easier. It doesn't require multiplicative weights.

(a) Justify that

$$\max_{\mathbf{p}} \min_{\mathbf{q}} V_R(\mathbf{p}, \mathbf{q}) \leq \min_{\mathbf{q}} \max_{\mathbf{p}} V_R(\mathbf{p}, \mathbf{q})$$

However, it's not immediately or intuitively clear that knowing your opponents strategy does not provide some advantage. It could be that there is some gap $\varepsilon > 0$ such that:

$$\max_{\mathbf{p}} \min_{\mathbf{q}} V_R(\mathbf{p}, \mathbf{q}) + \varepsilon = \min_{\mathbf{q}} \max_{\mathbf{p}} V_R(\mathbf{p}, \mathbf{q})$$

Therefore to finish the proof we'll have to prove that there can be no such ϵ. To do so, it would suffice to find some strategy $\mathbf{q}^*$ such that

$$\max_{\mathbf{p}} V_R(\mathbf{p}, \mathbf{q}^*) < \max_{\mathbf{p}} \min_{\mathbf{q}} V_R(\mathbf{p}, \mathbf{q}) + \varepsilon$$

and we will use multiplicative weights to construct such a strategy. To simplify our analysis, let's consider the restricted case of games in which all payoffs R_{ij} are either 1 or 0: intuitively where the row player either wins or loses, with nothing in between.

(b) Suppose we play the game repeatedly. Come up with an appropriate set of "experts" that we can use with the randomized weighted majority algorithm to determine what strategies to play.

(c) As the column player, we would consider an expert to make a mistake if it were to let the row player win, or obtain a payoff of 1. Under this interpretation, what are the total mistakes and error rate of our algorithm in terms of the game?

(d) Now what is the optimal error rate in terms of the game?

(e) Therefore, how can we bound the error rate of our algorithm?

(f) Now complete the proof of the Minimax Theorem.

Further Review

1. **(A Tight Analysis)** In lecture we saw that the simple procedure that multiplied the weight of each expert by $\frac{1}{2}$ whenever the expert made a mistake, resulted in

$$m = \text{ \#mistakes of algorithm } \leq 2.41(M + \log_2 n),$$

where M = #mistakes made by the best expert and n = # of experts. In Problem 1, we saw what happens if we replace 1/2 with 2/3. In general, it turns out that the closer this factor gets to 1, the better the bound. Show that if we reduce the weights by a factor of $(1-\epsilon)$ for $\epsilon \leq 1/2$, then the number of mistakes is

$$m \leq 2(1+\epsilon)M + O\left(\frac{\log n}{\epsilon}\right).$$

2. **(Experts with fractional loss)** In lecture we saw the randomized weighted majority algorithm, which scales the weight by $(1-\epsilon)$ when an expert makes a mistake. We bound the number of mistakes we make with the number of mistakes the best expert makes. Here, we are interested in generalizing this framework.

First, we will allow more than binary outcomes, so the experts are predicting from a set of possible outcomes. Then, instead of just being right or wrong, an expert's prediction can be valued from 0 to 1 (where 0 could mean a perfect prediction and 1 the worst prediction, with other values in between). We call this value the "loss", which generalizes the "mistakes" from our original framework. Once again, this is the quantity that we want to minimize.

Let $\mathcal{P}$ be all the possible outcomes. We define a matrix $M_{i,j}$, with $i \in [n], j \in \mathcal{P}$ to be the loss that expert i experiences when the outcome is j. For all i, j, we have $M_{i,j} \in [0,1]$. Similar to the algorithm in class, we initialize the weight of each expert to 1. To make a prediction, we randomly sample an expert with the weight. Our expected loss could be measured by summing over the expected loss of each round.

To use the loss matrix to update the weights, if the outcome of round t is j_t, for each expert, $w^{(t+1)} = w^{(t)}(1-\epsilon)^{M_{i,j_t}}$. Intuitively, expert i tends to make a decision that incurs M_{i,j_t} loss when the outcome is j_t.

Our expected loss each round, given that the outcome is j_t, is

$$\left(\sum_{i=1}^{n} w_i^{(t)} M_{i,j_t}\right) \Big/ \sum_{i=1}^{n} w_i^{(t)}$$

Let this be denoted by $M(E^t, j_t)$. We are interested in upper bounding $\sum_{t=1}^{T} M(E^t, j_t)$.

Let $\epsilon < \frac{1}{2}$. After T rounds, for any expert i, we want to show that

$$\sum_{t=1}^{T} M(E^t, j_t) \leq \frac{\ln n}{\epsilon} + (1+\epsilon)\sum_{t} M_{i,j_t}$$

Use these inequalities:

$$
\begin{aligned}
(1-\epsilon)^x &\leq (1-\epsilon x) && \forall x \in [0,1] \\
(1+\epsilon)^{-x} &\leq (1-\epsilon x) && \forall x \in [-1,0] \\
\ln\left(\frac{1}{1-\epsilon}\right) &\leq \epsilon + \epsilon^2 && \forall \epsilon : 0 < \epsilon < \frac{1}{2} \\
\ln(1+\epsilon) &\geq \epsilon - \epsilon^2 && \forall \epsilon : 0 < \epsilon < \frac{1}{2}
\end{aligned}
$$

and a similar potential function approach as in class to prove the bound above.