CS 6840: Algorithmic Game Theory **Spring 2017**

Lecture 12: February 22

Lecturer: Éva Tardos *Scribe: Ruoyun Chen*

12.1 Overview

We were talking about online learning. There is this kind of algorithm:

In each step 1, 2, ..., T

Choose one strategy a^t

Experience cost $c^t(a^t)$ at time t, we'll assume $0 \leq c^t(a^t) \leq 1$ for all time steps t and all strategies a^t.

Overall cost is $\sum_t c^t(a^t)$

And we hope to compare it with $\min_{a^*} \sum_t c^t(a^*)$, where a^* is the fixed action with hindsight

Using information about $c^k(a)$ for $k < t$, $a \in S$ (Full information).

What we will show today is an algorithm that derive the expected cost that gives the following inequality:

$$\mathbb{E}[\sum_{t=1}^{T} c^t(a^t)] \leq (1+\varepsilon) \min_{a^*} \sum_t c^t(a^*) + O(\frac{log \mid S \mid}{\varepsilon})$$

Recall: if all players use such algorithm, it results in a sequence of strategy vectors, $S^1, S^2, ...S^T$. If the probability distribution $\bar{\sigma}$ randomly chooses from this sequence, it results in approximately coarse correlated equilibrium (CCE).

Preview: Friday: From CCE to Nash Equilibrium. We will see in 2-player-zero-sum-game, "learning" converges to Nash; Monday: from Full-Info learning to Partial Info learning. We will see if we have a full-info algorithm, what we do for a partial-info situation.

12.2 The Multiplicative Weights Algorithm

Reminder:

- The algorithm need to be randomize to avoid being predictable for the advisory;
- The simplest randomization: choose uniformly random all the times. $\mid S \mid = n$, choose each with probability $\frac{1}{n}$.
  - The mathematical criticism: in bad case, $c^t(a^*) = 0$ for all t, all $c^t(a) = 1$ for all $a \neq a^*$. In this case, error grows with time;
  - The intellectual criticism: There is no learning in this algorithm.

12.2.1 Algorithm

So the algorithm starts with uniformly random. After iteration t, if $c^t(a)$ is small, then $p^t(a)$ goes up; if $c^t(a)$ is big, then $p^t(a)$ goes down.

But how to define "small" and "big" in this case? We will normalize to make $0 \leq c^t(a) \leq 1$.

We will then maintain $w^1(a) = 1$ for every $a \in A$, $W^t = \sum_a w^t(a)$, and the probability of choosing a is defined as $p^t(a) = \frac{w^t(a)}{W^t}$, so that the probability is guaranteed to sum up to 1.

As we proposed above, we want the probability of choosing one action decreases if its cost is big, and the probability of choosing one action increases if the cost is small. We therefore define the following adjustment based on the cost after each iteration:

$$w^{t+1}(a) = w^t(a)(1 - \varepsilon c^t(a))$$

Given the defined adjustment, we can see that

- When the cost of a is big at iteration t, we scale down the $w^{t+1}(a)$;
- When the cost of a is small, $w^{t+1}(a)$ is not scaled up. But implicitly we increase the $p^{t+1}(a)$ as we renormalize $w(a)$ to get $p(a)$.

Notice that ε is referred to a learning rate: how fast you're learning something is bad/good.

Therefore, **the Multiplicative Weights Algorithm:**

Initialize $w^1(a) = 1$ for every $a \in A$
for each time step $t = 1, 2, ..., T$ **do**
  use the distribution $p^t(a) = w^t(a)/W^t$ over actions $a \in S$,
    where $W^t = \sum_{a \in A} w^t(a)$ is the sum of weights
  given the cost vector c^t, for every action $a \in A$
    use the formula $w^{t+1}(a) = w^t(a)(1 - \varepsilon c^t(a))$ to update its weight

12.2.2 Proof the Expected Regret

Proof: Let's start with the upper bound of W^{t+1}.

$$W^{t+1} = \sum_a w^t(a)(1 - \varepsilon c^t(a)) = W^t(1 - \varepsilon \sum_a \frac{w^t(a)}{W^t} c^t(a)) = W^t(1 - \varepsilon \sum_a p^t(a)c^t(a)),$$

where $\sum_a p^t(a)c^t(a)$ is algorithm's expected cost.

Let $A^t = \sum_a p^t(a)c^t(a)$ be the expected cost at time t, then

$$W^{t+1} = W^t(1 - \varepsilon A^t) = w^1 \prod_{\tau=1}^{t}(1 - \varepsilon A^\tau) = n \prod_{\tau=1}^{t}(1 - \varepsilon A^\tau)$$

Given that

$$1 - x \leq e^{-x}$$

We have,

$$1 - \varepsilon A^t \leq e^{-\varepsilon A^t}$$

$$W^{t+1} = n \prod_{\tau=1}^{t} (1 - \varepsilon A^\tau) \leq n e^{-\varepsilon \sum_{\tau=1}^{t} A^\tau}$$

Next, let's look at the lower bound of W^{t+1}

$$W^{t+1} = \sum_a w^{t+1}(a) \geq w^{t+1}(a^*) = \prod_{\tau=1}^{t} (1 - \varepsilon c^\tau(a^*))$$

We are looking for the relationship that holds like

$$1 - x \geq e^{??}$$

We are then inspired by Taylor Series, which gives

$$ln(1-x) = -x - \frac{1}{2}x^2 - \frac{1}{3}x^3 - ...$$

From this, we have $ln(1-x) \geq -x - x^2$ if $x < \frac{1}{2}$

That is, $1 - x \geq e^{-x-x^2}$

Thus, we can rewrite the lower bound to

$$w^{t+1} \geq \prod_{\tau=1}^{t} e^{-\varepsilon c^\tau(a^*) - \varepsilon^2 c^\tau(a^*)^2} = e^{-\varepsilon \sum_{\tau=1}^{t} c^\tau(a^*) - \varepsilon^2 \sum_{\tau=1}^{t} c^\tau(a^*)^2}$$

which can be further relaxed, using that $0 \leq c^t(a) \leq 1$ for all a and all t, to

$$e^{-\varepsilon \sum_{\tau=1}^{t} c^\tau(a^*) - \varepsilon^2 \sum_{\tau=1}^{t} c^\tau(a^*)^2} \geq e^{-\varepsilon \sum_{\tau=1}^{t} c^\tau(a^*) - \varepsilon^2 \sum_{\tau=1}^{t} c^\tau(a^*)} \geq e^{-\varepsilon \sum_{\tau=1}^{t} c^\tau(a^*)(1+\varepsilon)}$$

Combining the lower bound and the upper bound,

$$e^{-\varepsilon \sum_{\tau=1}^{t} c^\tau(a^*)(1+\varepsilon)} \leq n e^{-\varepsilon \sum_{\tau=1}^{t} A^\tau}$$

Take log,

$$-\varepsilon \sum_{\tau=1}^{t} c^\tau(a^*)(1+\varepsilon) \leq ln(n) - \varepsilon \sum_{\tau=1}^{t} A^\tau$$

which is,

$$-ln(n) + \varepsilon \sum_\tau A^\tau \leq \varepsilon \sum_\tau c^\tau(a^*)(1+\varepsilon)$$

$$\sum_\tau A^\tau \leq (1+\varepsilon) \min_{a^*} \sum_\tau c^\tau(a^*) + \frac{ln(n)}{\varepsilon}$$

Recall that A^τ is the algorithm's expected cost at time τ. ■