No Regret and Correlated Equilibria

In this lecture, our goal is to see how correlated and coarse correlated equilibria can be found by using no-regret algorithms such as polynomial weights. We will see that in the case of correlated equilibrium, we need a new notion called “swap regret”.

First, let's see what happens if all players in a game are using a “no regret” algorithm such as polynomial weights. Recall:

Definition 1 *Given an n-player game, we say a sequence of action profiles $a^1, \ldots, a^T$ has* ***average regret*** *$\Delta(T)$ if for all i, player i has average regret at most $\Delta(T)$ for playing her sequence of actions $a_i^1, \ldots, a_i^T$. That is, for all deviations a_i',*

$$\frac{1}{T}\sum_{t=1}^{T} u_i(a_i', a_{-i}^t) \;-\; \frac{1}{T}\sum_{t=1}^{T} u_i(a^t) \leq \Delta(T).$$

(Note the slight difference from the notion of regret in previous lectures: Previously, the players wanted to minimize loss, whereas now they want to maximize utility.)

Theorem 2 *If $a^1, \ldots, a^T$ has average regret $\Delta(T)$, then the distribution $\mathcal{D}$ that uniformly at random picks one of these action profiles $a^1, \ldots, a^T$ is an (approximate) coarse correlated equilibrium. The approximation is $\Delta(T)$, meaning that every player can improve by at most $\Delta(T)$ by deviating.*

Proof If player i follows the recommendation a that is drawn from $\mathcal{D}$, her expected utility is

$$\mathop{\mathbb{E}}_{a\sim\mathcal{D}} u_i(a) = \sum_{t=1}^{T} \frac{1}{T} u_i(a^t).$$

This uses that each a^t is chosen with probability $\frac{1}{T}$.

If i deviates to a_i', she obtains expected utility

$$\mathop{\mathbb{E}}_{a\sim\mathcal{D}} u_i(a_i', a_{-i}) = \sum_{t=1}^{T} \frac{1}{T} u_i(a_i', a_{-i}^t).$$

By definition of the average regret of the sequence, she improves by at most $\Delta(T)$ by deviating. ■

Now, suppose we play the game repeatedly T times, where each player uses the polynomial weights algorithm to choose her actions at each time, based on the previous time steps.[1] Then by the regret guarantee of the polynomial weights algorithm, each player i guarantees expected regret at most $2\sqrt{\log|A_i|/T}$. If we let $k = \max_i |A_i|$, then every player i has expected regret at most $2\sqrt{\log(k)/T}$, so the sequence $a^1, \ldots, a^T$ has average regret at most $\Delta(T) = 2\sqrt{\log(k)/T}$.

This proves the following corollary.

Corollary 3 *By running the polynomial weights algorithm for all players, over T repeated rounds, we can compute an $\Delta(T)$-approximate coarse correlated equilibrium for $\Delta(T) = 2\sqrt{\log(k)/T}$, where k is the maximum number of actions of any player.*

[1] We technically need to first transform the game into an input the polynomial weights algorithm can handle, i.e. so that every outcome for each action is a loss in $[0, 1]$ rather than a utility. We can do so by scaling and shifting all utilities by the same amount, then using the negative of utility as the loss for the algorithm.

So far so good for *coarse* correlated equilibria. Are there learning algorithms that efficiently converge to correlated equilibrium? A natural approach (by analogy to how we can find coarse correlated equilibria) is to try and find an experts algorithm that has the following guarantee:

Definition 4 *Given an n-player game, we say a sequence of action profiles $a^1, \ldots, a^T$ has **average swap regret** $\Delta(T)$ if for all players i and all $f : A_i \to A_i$,*

$$\left(\frac{1}{T}\sum_{t=1}^{T} u_i(f(a_i^t), a_{-i}^t)\right) - \frac{1}{T}\sum_{t=1}^{T} u_i(a^t) \leq \Delta(T).$$

To distinguish this notion, we may sometimes refer to our previous definition of regret as "external regret".

Now we have:

Theorem 5 *If $a^1, \ldots, a^T$ have average swap regret $\Delta(T)$, then $\mathcal{D}$ that picks among them uniformly is a $\Delta(T)$-approximate correlated equilibrium.*

The proof is exactly analogous to the case of coarse correlated equilibrim.

But: how can we compute a sequence with low average swap regret? The polynomial weights algorithm doesn't necessarily guarantee good swap regret.

We recall/rephrase the online learning setting. Suppose there are k actions $j = 1, \ldots, k$, and each round t, we observe a loss ℓ_j^t for each action j. The algorithm must pick some action, call it j^t at time t, and receives a loss $\ell_A^t := \ell_{j^t}^t$. The *average swap regret* of the algorithm is

$$\frac{1}{T}\sum_{t=1}^{T} \ell_A^t - \min_{f:A_i \to A_i} \frac{1}{T}\sum_{t=1}^{T} \ell_{f(j^t)}^t.$$

The algorithm is:

1. Initialize k copies of the PW algorithm, one for each action $j = 1, \ldots, k$.
2. At each time t, the copy of PW for action j specified a distribution over actions, call it q_j^t. Note that q_j^t is a probability distribution over all k actions.
3. Combine (in a way described below) all of these distributions into a single distribution p^t on actions. Draw the action j^t to play from this distribution, where j is drawn with probability $p^t(j)$.
4. The losses $\ell_1^t, \ldots, \ell_k^t$ for the actions arrive. However, we do not directly give these losses to all the copies of the PW algorithm. Instead, the copy for action j receives all these losses scaled down by $p^t(j)$. That is, it receives the losses $p^t(j)\ell_1^t, \ldots, p^t(j)\ell_k^t$.

The above algorithm is fully specified, except for how we combine the k PW distributions $q_1^t, \ldots, q_k^t$ into a single distribution over actions p^t. We do so as follows. We set this distribution to satisfy the following system of equations: for each j,

$$p^t(j) = \sum_{j'=1}^{k} p^t(j') q_{j'}^t(j)$$

The above set of equations have a unique solution (note that there are k linear equations in k unknowns).

How should you think about the solution to this system? It is saying that, once you solve for p^t, the following two ways of picking an action are equivalent:

1. We draw an action j according to p^t. That is, each action j is chosen with probability $p^t(j)$.

2. We select a copy j' of the PW algorithm according to p^t. Then we draw an action j according to that copy of the PW algorithm, i.e. j is chosen with probability $q^t_{j'}(j)$.

Theorem 6 *The above algorithm achieves, in expectation, average swap regret at most* $2k\sqrt{\log(k)/T}$.

Proof Let $f : \{1, \ldots, k\} \to \{1, \ldots, k\}$ be any swap function.

We consider the jth copy of PW. Its expected loss at round t is

$$
\begin{aligned}
L^t_{A(j)} &:= \sum_{j'=1}^{k} (\text{probability it places on action } j')\,(\text{loss it observes for } j') \\
&= \sum_{j'=1}^{k} q^t_j(j')\left(p^t(j)\ell^t_{j'}\right) \\
&= p^t(j)\sum_{j'=1}^{k} q^t_j(j')\ell^t_{j'}.
\end{aligned}
$$

Now, by the guarantee of the polynomial weights algorithm, it has low (external) regret. In particular, it does almost as well as playing the action $f(j)$ every round:

$$
\frac{1}{T}\sum_{t=1}^{T} L^t_{A(j)} \le \frac{1}{T}\sum_{t=1}^{T} p^t(j)\ell^t_{f(j)} \;+\; \Delta(T)
$$

where, recall, $\Delta(T) = 2\sqrt{\log(k)/T}$. Now, let us sum both sides over all copies j of the PW algorithm:

$$
\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} L^t_{A(j)} \le \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p^t(j)\ell^t_{f(j)} \;+\; k\Delta(T). \tag{1}
$$

But what is $\sum_{j=1}^{k} L^t_{A(j)}$?

$$
\begin{aligned}
\sum_{j=1}^{k} L^t_{A(j)} &= \sum_{j=1}^{k} p^t(j)\sum_{j'=1}^{k} q^t_j(j')\ell^t_{j'} \\
&= \sum_{j'=1}^{k} \ell^t_{j'}\sum_{j=1}^{k} p^t(j)q^t_j(j') \\
&= \sum_{j'=1}^{k} p^t(j')\ell^t_{j'}.
\end{aligned}
$$

Here, we used our definition of $p^t(j)$. So Equation 1 becomes

$$
\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p^t(j)\ell^t_j \le \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} p^t(j)\ell^t_{f(j)} \;+\; k\Delta(T).
$$

The left side is the average loss of the algorithm. The right side is the average loss the algorithm would have suffered, in expectation, by playing $f(j)$ instead of j every time it had played j, for all actions j. Therefore, this shows that the average swap regret is bounded, in expectation, by

$$
k\Delta(T) = 2k\sqrt{\frac{\log(k)}{T}}.
$$

■