8.1 Introduction

Today we're going to finish the first part of the class, on computing equilibria, by showing that there are learning algorithms which lead to dynamics in which the time-averaged distribution of play converges to a correlated equilibrium. We already showed this for coarse correlated equilibria, but today we're going to go all the way to correlated equilibria. Next week we'll begin the second part of the course, on inefficiency of equilibria.

8.2 Correlated Equilibria and Swap Regret

Remember from last class we proved that a new definition of correlated equilibrium is equivalent to our original definition:

Definition 8.2.1 *σ is a correlated equilibrium if for all $i \in [k]$ and for all $\delta : S_i \to S_i$,*

$$\mathop{\mathbf{E}}_{s\sim\sigma}[C_i(s)] \leq \mathop{\mathbf{E}}_{s\sim\sigma}[C_i(s_{-i}, \delta(s_i))].$$

This was nice because it removed the conditioning part of the definition, but at the price of quantifying over all switching functions.

8.2.1 Relationship to Swap Regret

This new definition of correlated equilibria will be useful because it will naturally relate back to a concept that was studied (before anyone knew this connection!) in online learning theory known as *swap regret*. To motivate this, let's remember why we came up with our original notion of regret: we wanted an online learning algorithm with provable guarantees, but we proved an impossibility result which said that we couldn't hope to compete with the best sequence of actions (in hindsight). So instead we decided to try to compete with the best *single* action in hindsight. Another way of saying this is the following: go back in time, and for every time t consider what would have happened if we had switched from the action a^t that we chose to play then to instead play action a. Our regret $R_T(a)$ is precisely how much better off we would have been if we had done that: $\sum_{t=1}^{T} c^t(a^t) - \sum_{t=1}^{T} c^t(a)$. We saw that we can get no regret with this notion of regret, i.e., we can indeed compete with the best action in hindsight.

So we can't compete with the best sequence of action, but we can compete with the best single action, i.e., with what would have happened if we had always switched to play that action. A natural question is whether there things in the middle to compete with: notions of OPT that are stronger than the best single action, but which we can still actually compete against. This has been an active area of study in online learning for a long time, and (possibly surprisingly) the answer is a definitive yes. In fact, one of they key definition is what is known as *swap regret*: what if instead of competing with the best single action, we tried to compete with the best "set of swaps". In other words, we go back in time and whenever we used to play action a we instead play b, and whenever

we used to play b we instead play d, etc. More formally:

Definition 8.2.2 *The* swap regret *of a sequence of actions $a^1, a^2, \ldots, a^t$ with respect to a switching function $\delta : A \to A$ is*

$$S_T(\delta) = \frac{1}{T}\left(\sum_{t=1}^{T} c^t(a^t) - \sum_{t=1}^{T} c^t(\delta(a^t))\right)$$

Note that swap regret generalizes our old notion of regret, since $R_T(a) = S_T(\delta)$ where $\delta(x) = a$ for all $x \in A$. So if we have low swap regret (for all δ) then we definitely have low regret (for all a), but if we have low regret we might still have high swap regret. In other words: we're competing against a broader set of things in hindsight, so low swap regret is a stronger guarantee than low regret. But note that swap regret is not broad enough to include the actual best action sequence in hindsight (since we still have the restriction that in any two times when we played the same action a, we will have to play the same action after swapping).

As before, we can extend this definition to the expected swap-regret of an algorithm:

Definition 8.2.3 *Let $\mathcal{A}$ be an online learning algorithm. Then its expected swap regret with respect to $\delta : A \to A$ is*

$$\mathbf{E}[S_T^{\mathcal{A}}(\delta)] = \frac{1}{T}\left(\sum_{t=1}^{T} \mathop{\mathbf{E}}_{a^t \sim p^t}[c^t(a^t)] - \sum_{t=1}^{T} \mathop{\mathbf{E}}_{a^t \sim p^t}[c^t(\delta(a^t))]\right)$$

Note that by linearity of expectations, this is the same as the (possibly more obvious) definition of

$$\frac{1}{T}\sum_{t=1}^{T}\left(\mathop{\mathbf{E}}_{a^t \sim p^t}[c^t(a^t) - c^t(\delta(a^t))]\right)$$

And we can similarly define a no-swap-regret algorithm.

Definition 8.2.4 *An algorithm $\mathcal{A}$ has no-swap-regret if $\mathbf{E}[S_T^{\mathcal{A}}(\delta)] = o(1)$ as $T \to \infty$ for all $\delta : A \to A$.*

And now we can basically replicate the argument from no-regret to coarse correlated equilibria to prove that no-swap-regret algorithms give correlated equilibria. I'll run through this quickly since it's almost identical to what we did for no-regret and coarse correlated equilibria.

As before, let $\mathcal{A}_i$ be the algorithm used by player i, let p_i^t be the mixed strategy used by player i at time t (i.e., the distribution defined by $\mathcal{A}_i$), let $\sigma^t = \prod_{i=1}^{k} p_i^t$, and let $\sigma = \frac{1}{T}\sum_{i=1}^{T} \sigma^t$.

Theorem 8.2.5 *Suppose that $\mathbf{E}[S_T^{\mathcal{A}_i}(\delta)] \leq \epsilon$ for all players $i \in [k]$ and $\delta : S_i \to S_i$. The σ is an ϵ-approximate correlated equilibrium, in the sense that*

$$\mathop{\mathbf{E}}_{s \sim \sigma}[C_i(s)] \leq \mathop{\mathbf{E}}_{s \sim \sigma}[C_i(s_{-i}, \delta(s_i))] + \epsilon$$

for all $i \in [k]$ and for all $\delta : S_i \to S_i$.

Proof: Let $i \in [k]$ and let $\delta : S_i \to S_i$. Recall that the cost vector that player i sees at time t is $c^t(a) = \mathbf{E}_{s\sim\sigma^t}[C_i(s_{-i}, a)]$, and hence $\mathbf{E}_{a^t\sim p^t}[c^t(a)] = \mathbf{E}_{s\sim\sigma^t}[C_i(s)]$. Then we have that

$$\begin{aligned}
\mathop{\mathbf{E}}_{s\sim\sigma}[C_i(s)] - \mathop{\mathbf{E}}_{s\sim\sigma}[C_i(s_{-i}, \delta(s_i))] &= \frac{1}{T}\sum_{i=1}^{T} \mathop{\mathbf{E}}_{s\sim\sigma^t}[C_i(s)] - \frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbf{E}}_{s\sim\sigma^t}[C_i(s_{-i}, \delta(s_i))] \\
&= \frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbf{E}}_{a^t\sim p^t}[c^t(a^t)] - \frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbf{E}}_{a^t\sim p^t}[c^t(\delta(a^t))] \\
&= \mathbf{E}[S_T^{\mathcal{A}_i}(\delta)] \\
&\leq \epsilon
\end{aligned}$$

as claimed. ∎

So if all players use no-swap-regret algorithms, then the distribution of play starts to look like a correlated equilibrium!

8.3 No-Swap-Regret Algorithms

Now we want to actually design a no-swap-regret algorithm. To do this, we won't start from scratch like we did for a no-regret algorithm. We'll instead prove a result due to Blum and Mansour [BM07]: we can transform no-regret into no-swap-regret!

Theorem 8.3.1 ([BM07]) *If there is a no-regret algorithm, then there is a no-swap-regret algorithm.*

We're going to spend the rest of class proving this, or at least giving a sketch of the proof. Let's first set some notation. Let $A = [n]$, and let $M_1, M_2, \ldots, M_n$ be no-regret algorithms (multiplicative weights or something else). Note that these could all be different instantiations of the same algorithm (e.g., multiplicative weights). So we have one no-regret algorithm per action. Our combined algorithm, which we will prove to be no-swap-regret, will work as follows. At time $t = 1, 2, \ldots, T$:

- Receive distributions $q_1^t, q_2^t, \ldots, q_n^t$ from $M_1, M_2, \ldots, M_n$
- Compute "consensus" distribution p^t (this is the key step – as we go, we'll derive exactly how to do this to get the guarantees that we want).
- Receive c^t from adversary and pay appropriately.
- Give M_j the cost vector $p^t(j)c^t$, i.e., "lie" to algorithm M_j about the real cost vector by multiplying it through by $p^t(j)$.

The time-averaged expected cost of this algorithm is

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} p^t(i)c^t(i). \tag{I}$$

Consider some switching function $\delta : A \to A$. If we switched according to δ, then the time-averaged expected cost would be

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} p^t(i)c^t(\delta(i)). \tag{II}$$

So to prove no-swap-regret, we want to prove that (I) - (II) = $o(1)$.

Let's look at this process from the perspective if M_j. This algorithm is no-regret, and so in particular has no-regret with respect to $\delta(j)$. But this no-regret is with respect to its perceived cost vectors, not the true cost vectors. If we write this out more formally, we get:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} q_j^t(i)(p^t(j)c^t(i)) - \frac{1}{T}\sum_{t=1}^{T} p^t(j)c^t(\delta(j)) \le R_T^j(\delta(j)) \qquad (= o(1)),$$

where the first time is M_j's perceived cost and the second term is M_j's perceived cost in hindsight of always playing action $\delta(j)$.

Now let's sum this inequality over all $j \in [n]$ to get

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} q_j^t(i)(p^t(j)c^t(i)) - \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{n} p^t(j)c^t(\delta(j)) \le \sum_{j=1}^{n} R_T^j(\delta(j)) = o(1),$$

where for the last equality we just used the fact that our asymptotic notation is with respect to T, not n.[1] Look at the second term of the left hand side: it is precisely (II)! So if we can prove that the first term is equal to (I), then we'll be finished. In other words, we just want to prove that

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} q_j^t(i)p^t(j)c^t(i) = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n} p^t(i)c^t(i). \tag{8.3.1}$$

Note that we *still* haven't described how to actually choose p^t (the consensus distribution); everything until now has been true for arbitrary p^t. But now we're going to set p^t so that (8.3.1) becomes true. To do this, notice that (8.3.1) is definitely true if

$$p^t(i) = \sum_{j=1}^{n} q_j^t(i)p^t(j) \tag{8.3.2}$$

for all t and i, since then for every t and i the multiplier of $c^t(i)$ will be the same on both sides. This looks weird – we've kind of defined p^t in terms of itself. But it turns out that there's a pretty simple way of doing this. Let's define a Markov chain with states $A = [n]$ and with the transition probability from j to i equal to $q_j^t(i)$. Note that this is indeed a Markov chain, since $\sum_{i=1}^{n} q_j^t(i) = 1$ (since q_j is the output distribution of M_j). Since this is a finite Markov, there is always some stationary distribution. And any stationary distribution will exactly satisfy (8.3.2), and in fact (8.3.2) is precisely the definition of a stationary distribution!

[1] Slightly more formally, using multiplicative weights we get regret of $O\left(\sqrt{\frac{\log n}{T}}\right)$, so summing over the actions makes the total sum $O\left(\sqrt{\frac{n^2 \log n}{T}}\right)$, which is still $o(1)$ as T goes to ∞.

8.4 Conclusion

We've only scratched the surface of the computation of equlibria in general, and connections between machine learning and AGT. For anyone who's interested, there's a great (though somewhat old) book on precisely these connections by Cesa-Bianchi and Lugosi [CBL06]. And this is still an active area of research: a few years ago, with Raman Arora, Teodor Marinov, and Mehryar Mohri, we recently [ADMM18] investigated what happens if players use a definition of regret known as *policy regret* that Raman defined earlier [DTA12]. It turns out that players don't necessarily get to a type of equilibrium that had previous been defined, so we had to define a new type of equilibrium that we called a *policy equilibrium*. There are interesting connections between policy equilibria and coarse correlated equilibria, but there's still a lot that we don't understand.

And, of course, there are all sorts of games that we're not talking about (multistage, limited information, etc.). In all of these games there are classical notions of equilibria, and we can ask about the difficulty of computing them and whether "natural" dynamics converge to them (at least in a time-averaged way). There's a lot of work on this, and I could teach a whole course just on computing equilibria, but since we have only limited time we're going to stop here. These are great opportunities for course projects, though!

References

[ADMM18] Raman Arora, Michael Dinitz, Teodor Vanislavov Marinov, and Mehryar Mohri. Policy regret in repeated games. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 6733–6742, 2018.

[BM07] Avrim Blum and Yishay Mansour. From external to internal regret. *J. Mach. Learn. Res.*, 8:1307–1324, December 2007.

[CBL06] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, USA, 2006.

[DTA12] Ofer Dekel, Ambuj Tewari, and Raman Arora. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.