CS 6840 Algorithmic Game Theory November 1, 2024

Lecture 28: Algorithmic Learning and Collusion continued

Instructor: Eva Tardos *Scribe: Sophie Greenwood*

In this lecture, we continue examining algorithmic collusion.

1 Review from last lecture

Recall the example from last class, where there is 1 "unit" of buyers – that is, an infinitely large set of buyers with measure 1 – which each have value of greater than 1 for the product.

Suppose there are two sellers considering prices in $\left\{\frac{1}{2k}, \frac{2}{2k}, ..., \frac{2k}{2k}\right\}$.[1] Note that at any of these prices, all buyers will want to buy the product. All buyers will go to the cheaper seller and purchase the item from them. This setting is commonly referred to as *Bertrand competition*.

Suppose the two sellers charge prices p_1 and p_2 respectively. Then

$$u_1(p_1, p_2) = \begin{cases} 0, & p_1 > p_2 \\ p_1/2, & p_1 = p_2 \\ p_1, & p_1 < p_2 \end{cases}.$$

In the Nash equilibria, all prices are $1/2k$ or $2/2k$.

Last time, we showed that if one player is a no-regret learner and the other player is a Stackelberg leader, the outcome for both players is much better than the Nash equilibrium. In the Nash equilbrium, both players have utility $O(1/k)$; in the outcome with a Stackelberg leader and where the follower is a no-regret learner both players get $\Omega(1)$ utility.[2]

2 Both players are no-regret learners

In this section, we seek to answer the question: *What will happen if both sellers are no-regret learners?*

Last time, we considered a Stackelberg leader with a uniformly random policy, which put positive probability on $p = 1$. *Can a no-swap-regret learner use $p = 1$?*

Suppose that user i uses a strategy σ_i in which she uses price 1 with probability c_i.

- If player 1 uses price 1, she will have expected utility

$$\begin{aligned} \mathbb{E}_{p_2 \sim \sigma_2} u_1(1, p_2) &= \Pr(p_2 < 1) \cdot 0 + \Pr(p_2 = 1) \cdot [(1/2)1] \\ &= (1 - c_2) \cdot 0 + c_2 \cdot 1/2 \\ &= \frac{1}{2} c_2 \end{aligned}$$

[1] We take the denominator to be an even number ($2k$) for simplicity.

[2] Big-Ω is the lower-bound counterpart to big-O: the notation $g \in \Omega(f)$ means "g is at least on the order of f".

- If player 1 uses price $\frac{2k-1}{2k}$ (the next-highest price) instead, she will have expected utility

$$\begin{aligned}\mathbb{E}_{p_2\sim\sigma_2} u_1\left(\frac{2k-1}{2k}, p_2\right) &\geq \Pr\left(p_2 < 1\right)\cdot 0 + \Pr(p_2 = 1)\cdot\left(\frac{2k-1}{2k}\right)\\ &= (1-c_2)\cdot 0 + c_2\cdot\left(\frac{2k-1}{2k}\right)\\ &= \frac{2k-1}{2k}c_2.\end{aligned}$$

Therefore, choosing price 1 is strictly dominated by choosing price $\frac{2k-1}{2k}$. We know that after a sufficiently long time, the probability a no-swap-regret learner will choose a strictly dominated action approaches 0.

However, we can repeat this argument supposing that the highest price any seller will select is $p \leq 1$: suppose that user i uses strategy σ_i which chooses price p with probability c_i.

- If player 1 uses price p, she will have expected utility

$$\begin{aligned}\mathbb{E}_{p_2\sim\sigma_2} u_1(p, p_2) &= \Pr(p_2 < p)\cdot 0 + \Pr(p_2 = p)\cdot(1/2)p\\ &= (1-c_2)\cdot 0 + c_2(1/2)p\\ &= \frac{p}{2}c_2.\end{aligned}$$

- If player 1 uses price $p - \frac{1}{2k}$, she will have expected utility

$$\begin{aligned}\mathbb{E}_{p_2\sim\sigma_2} u_1\left(p-\frac{1}{2k}, p_2\right) &\geq \Pr\left(p_2 < p\right)\cdot 0 + \Pr(p_2 = p)\cdot\left(p-\frac{1}{2k}\right)\\ &= (1-c_2)\cdot 0 + c_2\left(p-\frac{1}{2k}\right)\\ &= \left(p-\frac{1}{2k}\right)c_2.\end{aligned}$$

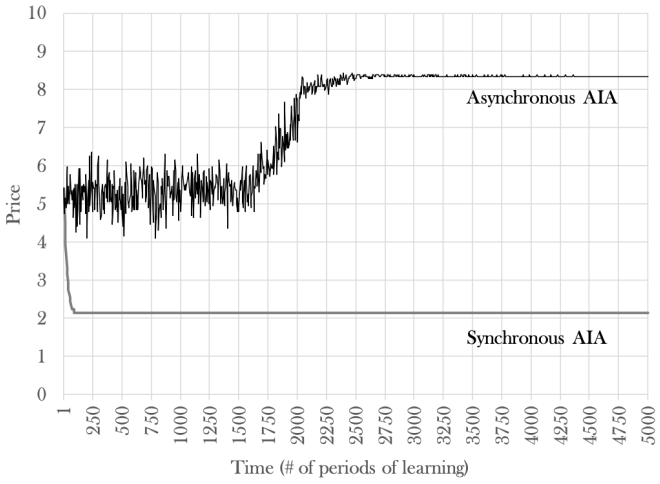
Now p is a strictly dominated choice, and over time p will not never be selected, making $p - 1/2k$ the highest price any seller will pay. By induction, we may continue this process until $p = 2/2k$.

This is true for any no-swap-regret algorithm; it's also true for for the multiplicative weights algorithm, since we showed on the homework that the probability of strictly dominated actions being played will tend to zero. Of course, a careful analysis requires a very careful treatment of error and regret, especially since the multiplicative weights algorithm never actually puts zero probability on events.

2.1 Learning to collude in simulations

However, this isn't always true in computational experiments. For example, in Figure 1 of Asker et al. [2022] (see Figure 1), the synchronous AIA (artificial intelligence algorithm) curve corresponds to an algorithm that updates like multiplicative weights, where you know not only the utility of your action, but the utility of other actions you might have taken – in this case, you know the price selected by the other player, and therefore can understand how your outcome would be different if you had chosen different prices. This is a "full information update". The asynchronous AIA curve corresponds to an algorithm that both does not do a full information update to its policies and *does not correct for* the fact that it only has partial information. This results in collusion: the asynchronous AIA algorithm converges to a price well above 2 (which is the marginal cost and hence the minimum price in this example; in general this setting is slightly different than ours).

Figure 1: Price paths with different algorithm designs



Notes: The median price (vertical axis) across 100 simulated runs by period (horizontal axis) is shown for firm 1, in a static Bertrand market in which two firms are selling homogeneous goods. The model is parametrized as follows. Market demand is $Q = 1$ if $P < 10$, zero otherwise. Marginal cost = 2. There are 100 feasible prices equally spaced between 0.1 and 10 inclusive. Firms put a zero weight on future profits. The model is parametrized as per figure 2. See notes therein for further details.

Figure 1: Figure 1 of Asker et al. [2022]

2.2 Learning to collude in the prisoner's dilemma

To flesh this out, consider the prisoner's dilemma game (with rewards/utilities rather than punishments/costs):

	C	D
C	$(5,5)$	$(0,6)$
D	$(6,0)$	$(1,1)$

Here C represents to cooperation and D represents to defection. Recall that in this game, (D, D) is the unique Nash equilibrium. Suppose that a learning algorithm is initialized like the multiplicative weights algorithm, with 50/50 probability on each action.

What happens if the algorithm doesn't know the structure of the game?

- The initial estimate is $u_C = u_D = 0$.
- If both players choose actions C and D with 50/50 probability, then there is a 1/8 chance that in the first two rounds, the outcomes are (C, C) and (D, D) (in either order).
- If this happens, both players now estimate $u_C = 5$, $u_D = 1$, and start collaborating more.
- If the two players play forever, they will *eventually* stop collaborating.
- However in reality, if players see an action with consistently bad outcomes, they will put zero probability on it – here, they would put zero probability on defection and always cooperate.

2.3 Q-learning

This type of algorithm that only updates based on what it has seen is called "Q-learning". This algorithm did not originate in game theory, but was intended for settings with a single player in a stateful environment. In general, the Q-learning update is

$$u'_C = (1-\alpha) \cdot u_C + \alpha \cdot r_C$$

where r_C is the reward for taking action C in the current iteration, u_C is the previous estimate of the reward for taking action C, and u'_C is the updated estimate.

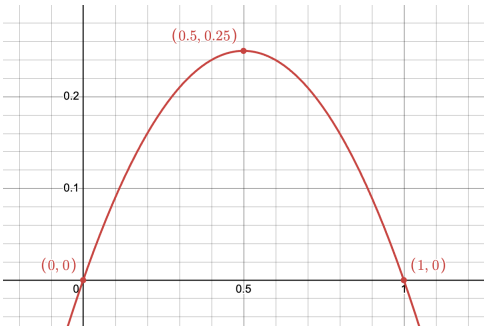


Figure 2: Plot of $p(1-p)$ vs. p.

2.4 Learning to collude in Bertrand competition

Now, we return to the Bertrand competition pricing setting.

Example. Assume players start with a uniformly random price p. Then the value of price p is $p \cdot 0 + (1-p) \cdot p = p - p^2$. This curve is visualized in Figure 2; the optimum is at $p = 1/2$. However, if $p \approx 0, p \approx 1$, the reward is quite low, so a learner might just stop playing those actions altogether.

We consider this in the next example.

Example. We now suppose that the players select a uniform price between $1/4$ and $3/4$. In this case, the expected utility of a player (without loss of generality, player 1) for selecting price p is

$$u_1(p) = \begin{cases} p, & p < 1/4, \\ 2x \cdot 0 + (1-2x) \cdot (1/4 + x), & 1/4 \le p := 1/4 + x \le 3/4, \\ 0, & p > 3/4. \end{cases}$$

The middle term is derived from the fact that when $p = 1/4 + x \le 3/4$ with probability $2x$, player 2's price is below p and player 1 has no customers. With probability $1-2x$, player 2's price is above p, and player 1 sells to all customers at price $p = 1/4 + x$.

We visualize the curve $(1-2x)(x+1/4)$ in Figure 3 The optimum is at $x = 1/8$, that is, $p = 1/4 + 1/8 = 3/8$. This optimal price is slightly lower than before ($p = 1/2$) but it is still quite high.

When we return to Figure 1, we see that initially, the price hovers around $1/2$. It's less clear why it jumps up over time.

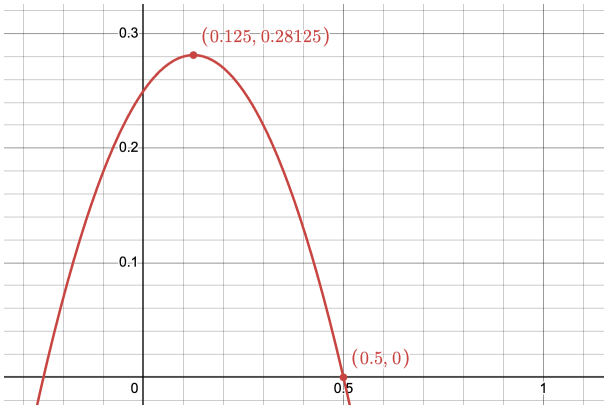


Figure 3: Plot of $(1-2x)(x+1/4)$ vs. x.

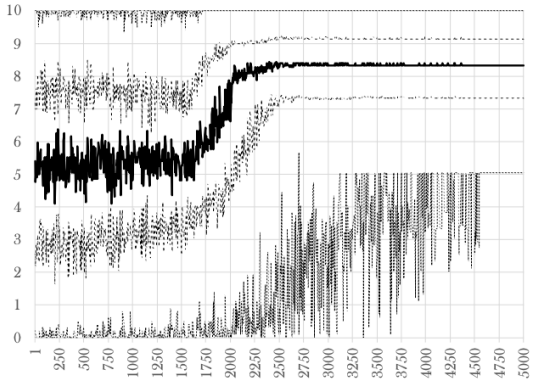


Figure 4: Figure 2b of Asker et al. [2022]

Asker et al. [2022] have other plots where they show individual runs rather than an average, such as Figure 4. The prices over time depend substantially on the price at the start of the run (note, these runs also show an upward jump.)

The speaker at Theory Seminar on 11/04 shows a similar result in a pricing setting with only two prices: he shows that the learned price can be the higher price. All of these results are for algorithms using Q-learning. To reiterate, Q-learning is intended for an environment with *states*; in this case, it estimates the expected utility over time of taking action C in state x for each C and x, assuming the environment is static. The reason the update is not a pure average but uses the term α is that you are simultaneously learning the expected reward at that step and what the expected future reward will be, and the estimates of the future rewards are improving. There are theoretical results which establish a basis for this.

References

John Asker, Chaim Fershtman, and Ariel Pakes. Artificial intelligence, algorithm design, and pricing. *AEA Papers and Proceedings*, 112:452–56, 2022. URL `https://www.aeaweb.org/articles?id=10.1257/pandp.20221059`.