CS 6840 Algorithmic Game Theory May 6, 2020

Lecture 33: Learning in Stackelberg Equilibrium

Instructor: Eva Tardos *Scribe: Sudeep Salgia*

In the previous lecture, we saw the concept of Stackelberg equilibrium that arises from the dynamics of a game where one player is the leader and the other is a learner (or equivalently a follower). In this lecture, we briefly recap the idea and then explore the settings where the leader can leverage the knowledge of the no-regret algorithm being used by the learner to have a value significantly larger than the Stackelberg value. We conclude with a brief discussion of what implications this might have for the learner.

Stackelberg Equilibrium

Consider the game given by the following payoff matrix

		Player 2 L	Player 2 R
Player 1	U	1, 1	0, 3
	D	0, 0	1, 2

In the above game, Player 1 is the Stackelberg leader while Player 2 is the learner. It is not difficult to see that the above game has a unique Nash equilibrium given by the strategy pair (U, L). As discussed in the previous lecture, the best Stackelberg strategy for the leader is to play each U and D roughly half the time (with D slightly more than U). This is because it would make R the best strategy for the learner, leading to a payoff of approximately 2.5 for the leader. It is clear that this state, referred to as the Stackelberg equilibrium, is much better for the leader than the Nash equilibrium.

The Stackelberg equilibrium can formally be defined as follows. Let $\boldsymbol{\alpha} = \{\alpha_i\}_{i \in S}$ be a probability distribution over S, the set of possible strategies for player 1 (the leader). Under this probability distribution $\boldsymbol{\alpha}$, the best response for player 2 (the learner) is given by $j(\boldsymbol{\alpha}) = \arg\max_k \sum_{i \in S} \alpha_i u_2(i, k)$. The Stackelberg equilibrium is given by the probability distribution $\boldsymbol{\alpha}$ that maximizes the value $\sum_{i \in S} \alpha_i u_1(i, j(\boldsymbol{\alpha}))$.

The concept of Stackelberg equilibrium is popular in the area of security games, where it is important for a security agency to protect a valuable resource from a potential threat. One of the prominent examples is that of airport security, where the security agency has to guard against any threat to the airport or the flights. Here, the security agency is the leader while the thief is the learner who will design the best response to the security measures, i.e., the strategy of the leader. Therefore, from the perspective of the security agency, the objective is to design a strategy that minimizes the damage to the resource. Another field where this has been applied is animal conservation, to fight against poaching.

An interesting insight is obtained when one considers Stackelberg equilibrium in zero sum games. As we have seen in the previous lectures that learning in zero sum games guarantees the learner at least the Nash value minus the regret incurred in learning. This implies that in zero sum games the Stackelberg leader does not have any advantage and the value that the leader will get cannot be better than Nash.

Beyond the Stackelberg Equilibrium

Consider the game given by the following payoff matrix

		Player 2		
		L	M	R
Player 1	U	ϵ, 0	-1, -2	0, -2
	D	-1, 0	1, -2	0, 2

where $\epsilon > 0$ is a small number. As before, player 1 is the leader while player 2 is the learner. It is not difficult to see that there are several possible Nash equilibria and again the strategy pair (U, L) is a Nash equilibrium. In that Nash equilibrium, the payoff of the players is 0 and ϵ respectively. Using the same procedure as for the previous problem, we can conclude that the Stackelberg value for the leader is 0 which is the same as Nash. Therefore, at this point, one might think it is not possible for the leader to do any better than this. Despite what may appear, we claim that the leader can do better (in fact, significantly better) than this if the leader knows the no-regret algorithm that the learner is using.

To prove the claim, let us analyze the situation from the perspective of the leader. As the leader, the only way we can get a positive value is if we force the learner to play the strategy R as often as possible. Consider the case where, as the leader, we play each of the strategies U and D with equal probabilities. In such a case, R is the best response for the learner but we get a payoff of 0. To force the learner to play R more often, we would have to play U more often. However, such a move would lead to a reduction to the payoff of the leader and hence it would do us more harm than good.

Now, consider the case where the leader knows that the learner is using Follow the Leader learning algorithm to learn. Given a time horizon T, the leader plays U for the first $T/2$ time steps. Since the learner is using Follow the Leader, it will play L for this period. The payoff for the leader in this period would be zero. Now, for the next $T/2$ time steps, the leader plays D. Since the leader knows the algorithm, it can predict what the learner would do. Let us a compute the cumulative payoff for each of the possible strategies of the learner after $x < T/2$ rounds of playing D, since that would be the guiding principle for choosing the strategy that the learner plays. The cumulative payoffs of the strategies L, M and R would be $\epsilon T/2 - x$, $-T/2 + x$ and 0 respectively. Note that these payoffs are ignoring the randomness. We are ignoring the aspect of randomization in this discussion for the sake of simplicity. With a little more involved argument, this can be extended to include the randomization. Note that the payoff for M is negative while that for R it is constantly 0. On the other hand, the payoff for strategy for L is initially positive but soon after x becomes greater than $\epsilon T/2$ it becomes negative. This would force the learner, who is using Follow the Leader, to play R for the rest of the time! Therefore, the payoff for the leader under this strategy is roughly $T(1 - \epsilon)$ since till $x = \epsilon T/2$ the payoff for the leader is 0 (learner plays L) and after that it is 2 (learner switches and starts playing R). This is certainly significantly larger than the 0 payoff that the leader was getting from the Stackelberg equilibrium.

It is interesting to note that even in this strategy, the leader effectively ended up playing U and D each for half the time, same as in the case of Stackelberg equilibrium. However, the difference here is that the strategies were not chosen at random at each instant. Instead, using the knowledge of the algorithm, the leader arranged the order of playing U and D to force learner to play R for almost half the time, i.e., when the leader was playing D. Thus, the leader leveraged the structure of the learning algorithm to get a better outcome for himself.

An immediate question that arises is whether the learner can prevent the leader to take advantage of him. More specifically, can the learner prevent the leader to get a value greater than his Stackelberg value? It is clear that the learner cannot do anything better than that. An important thing to notice is that in the above example the payoff of the learner remains almost unchanged. Therefore, the leader ended up gaining without affecting the payoff of the learner too much. However, in certain cases, like auction, the gain of the leader would come at the cost of the learner and in such cases, it would become imperative for the learner to ensure that the leader cannot do any better than the Stackelberg value.

In the next lecture, we look into the methods of how one can design algorithms to achieve the above. The fundamental idea behind the solution lies an alternative notion of regret, known as swap regret. In no-regret algorithm, the objective is to design policies to achieve $\sum_t u_i(s^t) \geq \max_{s_i^*} \sum_t u_i(s_i^*, s_{-i}^t)$. For no-swap regret framework, one needs to ensure that $\sum_{t:s^t=s_i'} u_i(s^t) \geq \max_{s_i^*} \sum_{t:s^t=s_i'} u_i(s_i^*, s_{-i}^t)$ holds for all $s_i' \in S$. The idea is that if this does not hold, then the algorithm made a mistake when it chose s_i' and should have swapped it with some other better performing strategy. Clearly, no swap regret is a stronger notion than no-regret and consequently, no swap regret implies no regret.

In fact, we can show that there is an adjustment to Follow the Perturbed Leader and Multiplicative Weight algorithms that can make them no-swap regret algorithms. The compromise is mainly in terms of speed as the modified algorithms are noticeably slower. However, there is only a little compromise on the error term in learning. More importantly, will prove a theorem in the next class which states that if the learner learns using a no swap regret algorithm, then the leader cannot have a value more than the Stackelberg value. Furthermore, we can also comment on the convergence in the context of learning with no swap algorithms. In a previous part of the course, we saw that if all the players play using a no-regret learning algorithm, then the empirical distribution of play converges to a coarse correlated equilibrium. We can now show that if all the players play using a no swap regret algorithm, then the empirical distribution converges to the correlated equilibrium. This relation adds another layer of intricacy between the coarse correlated equilibrium and correlated equilibrium.