CS6781 - Theoretical Foundations of Machine Learning

Lecture 22: Generalized FTPL

April 28, 2020

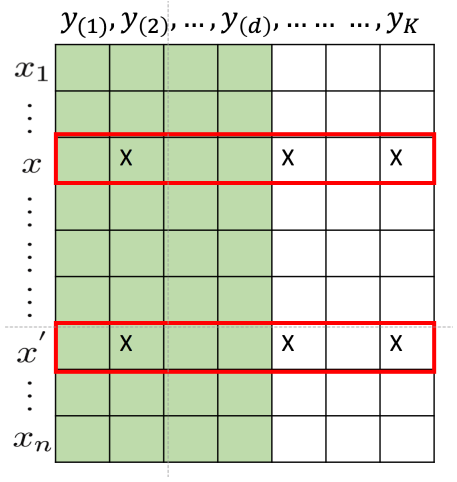
Lecturer: Nika Haghtalab *Readings: N/A*
Scribe: Richard Lanas Phillips

1 Generalized FTPL

In this lecture, we continue our study of oracle-efficient online learning. Recall that even given access to an offline 'oracle' that can solve a given problem with high accuracy, Hazan and Koren [2016] showed that online low-regret algorithms cannot exist in general with sub-exponential dependence on n. This means that there is a persistent exponential gap between online learning and offline learning in the general cases. Recent work has shown that certain assumptions on the structure of the problem can allow for oracle-efficient learning. Examples include FTPL for linear objective functions Kalai and Vempala [2005] and a generalization to submodular objective functions Hazan and Kale [2012]. Dudik et al. [2017] unifies these algorithms and presents a set of sufficient conditions for oracle-efficient online learning. The authors present *Generalized-Follow-the-Perturbed-Leader* for oracle-efficient learning based on the concept of d-structures. Recall from last lecture we have that

Definition 1.1. An online optimization problem with utility function $u : \mathcal{X} \times \mathcal{Y}$ is to have a d-structure if there are d actions of the adversary $y_{(1)}, \ldots, y_{(d)} \in \mathcal{Y}$, such that for any two different actions of the learners $x, x' \in \mathcal{X}$, there is $j \in [d]$ such that, $u(x, y_{(j)}) \neq u(x', y_{(j)})$.

We can understand the definition by the figure below using the game matrix. Existence of a d-structure means that there are d columns such that any two rows x and x' are distinct when we only consider the submatrix induces by columns in the d-structure.



Theorem 1.2. *If the problem has a d-structure and the utilities are 0-1, then you can have a no-regret algorithm with regret $O(d\sqrt{T})$ and oracle-efficient runtime of $poly(d, T)$.*[1]

We will analyze *Generalized FTPL* for oracle efficient learning when the problem has a d-structure. This is he algorithm that satisfies the requirements of Theorem 1.2. The algorithm takes the d-structure $\{y_{(1)}, \cdots, y_{(d)}\}$. For each of the adversarial actions $y_{(j)}$ in the d-structure, it draws a random number α_j, which is going to be the number of the copies of that action that will be added to the hallucinated history. This is a generalization of FTPL when perturbations look like changes to the actual history, instead of i.i.d random perturbation across the experts.

Algorithm 1 Generalized FTPL (Dudik et al. [2017])

1: Draw $\alpha_j \sim Unif[0, \sqrt{T}]$ independently for $j = 1, 2, 3, ...d$.
2: **for** $i = 1, ..., T$ **do**
3: Play

$$x^t = \text{OPT}\left(\{y^\tau\}_{\tau=1}^{t-1} \cup \bigcup_{j\in[d]} \alpha_j y_{(j)}\right)$$

4: Observe y^t and receive payoff $u(x^t, y^t)$.
5: **end for**

It is simple to see that the algorithm runs in $poly(d, T)$ and that it is oracle-efficient. What remains to be shown is that the above regret bound holds true. To show this, we will examine the performance on the real history plus the performance on the fake history in Equation 1. Note that the second term in this equation corresponds to the single random cost $c_0(x)$ in standard FTPL. Note that equivalently x^t can be defined by

$$x^t = \arg\max_x \left[\sum_{\tau=1}^{t-1} u(x, y^\tau) + \vec{\alpha} \cdot M_x\right] = \arg\max_x \left[\sum_{\tau=1}^{t-1} u(x, y^\tau) + \sum_{j=1}^{d} \alpha_j u(x, y_{(j)})\right]. \quad (1)$$

Recall the stability lemma for FTRL

$$\mathbb{E}[\text{Regret}] \le \mathbb{E}\left[\sum_{t=1}^{T} u\left(x^{t+1}, y^t\right) - u\left(x^t, y^t\right)\right] + \mathbb{E}\left[R(x_1) - R(x^*)\right], \quad (2)$$

which bounds the regret in terms of the stability and perturbation cost. Let M be the $|\mathcal{X}| \times d$ submatrix of the game matrix referring to columns $y_{(1)}, \ldots, y_{(d)}$ and let M_x be the row vector in this matrix referring to x. Then in this setting we have

$$\mathbb{E}[\text{Regret}] \le \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} u\left(x^{t+1}, y^t\right) - u\left(x^t, y^t\right)\right]}_{(1)} + \underbrace{\mathbb{E}\left[\vec{\alpha} \cdot (M_{x^1} - M_{x^*})\right]}_{(2)}. \quad (3)$$

[1]The assumption of 0-1 utilities is added for simplicity in thees lectures. Refer to [Dudik et al., 2017] to an unrestricted version of this theorem.

Note that (2) can be bounded as follows:

$$\mathbb{E}\left[\vec{\alpha}\cdot(M_{x^1}-M_{x^*})\right]\leq\mathbb{E}\left[\|\vec{\alpha}\|_\infty\cdot\|M_{x^1}-M_{x^*}\|_1\right]\leq O\left(d\sqrt{T}\right).$$

So it's sufficient to also bound (1) by $\mathcal{O}\left(d\sqrt{T}\right)$. For that it is sufficient to show that for a fixed t, $\Pr\left[x^{t+1}\neq x^t\right]\leq\mathcal{O}\left(d/\sqrt{t}\right)$. Using the uniqueness of the rows, we can state that

$$\Pr[x^{t+1}\neq x^t]\leq\sum_{i=1}^{d}\Pr[x^t\text{ and }x^{t+1}\text{ differ on }Col(i)\text{ of }M].$$

Regret bound Without loss of generality, let us consider $i=1$. For simplicity, we will focus on bounding the probability $\Pr[x^t$ has a 0 in column 1, x^{t+1} has 1 in col 1]. Below we'll define x^0 to be one of the best experts at time t among all those for which $M_{xi}=0$. And, x^1, one of the best experts at time $t+1$ among all those for which $M_{xi}=1$. That is,

$$\begin{aligned}x^0&\in\arg\max_{x:M_{x1}=0}\sum_{\tau=1}^{t-1}u\left(x,y^t\right)+\vec{\alpha}\cdot M_x\\&=\arg\max_{x:M_{x1}=0}\sum_{\tau=1}^{t-1}u\left(x,y^t\right)+\sum_{i=2}^{d}\alpha_i u\left(x,y^{(i)}\right)+\alpha_1\times 0.\end{aligned}$$

and

$$x^1\in\arg\max_{x:M_{x1}=1}\sum_{\tau=1}^{t}u\left(x,y^t\right)+\vec{\alpha}\cdot M_x=\arg\max_{x:M_{x1}=1}\sum_{\tau=1}^{t-1}u\left(x,y^t\right)+u(x,y^1)+\sum_{i=2}^{d}\alpha_i u\left(x,y^{(i)}\right)+\alpha_1\times 1.$$

Note that in the above equations α_1 does not play a role in the selection of x^0 and x^1, as it is a constant.

Let S' be the fake history up to time t excluding the first column of perturbations, that is, $S':=\{y_1,\ldots,y_{t-1},\alpha_2 y^{(2)}\ldots,\alpha_d y^{(d)}\}$. We'll also define

$$\Delta:=\text{Utility of any expert }x^0\text{ on set }S'-\text{Utility of any expert in }x^1\text{ on set }S'.$$

This prepares us to make the following useful observations.

1. Given that x^t has value 0 in column 1, then the utility of x^0 is higher than that of x^1 on on the actual perturbed history $S'\cup\alpha_1 y^{(1)}$. That is,

$$\Delta+\alpha_1 u(x^0,y^{(1)})-\alpha_1 u\left(x^1,y^{(1)}\right)\geqslant 0\implies\Delta\geq\alpha_1.$$

2. Given that x^{t+1} has value 1 in column 1, then the utility of x^1 is higher than that of x^0 on on the actual perturbed history at time $t+1$, $S'\cup\alpha_1 y^{(1)}\cup y^t$. That is,

$$\Delta+\alpha_1 u(x^0,y^{(1)})-\alpha_1 u\left(x^1,y^{(1)}\right)+u\left(x^0,y^{(t)}\right)-u\left(x^1,y^{(t)}\right)\leq 0\implies\Delta\leq\alpha_1+u(x^1,y^{(t)})-u(x^0,y^{(t)}).$$

Note that utility is bounded $[0, 1]$, therefore, $-1 \leq u(x^0, y^{(t)}) - y(x^1, y^{(t)}) \leq 1$. As a reminder, for our stability analysis we are interested in bounding the event $\Pr[x^t$ and x^{t+1} differ on $Col(i)$ of $M]$:

$$\Pr[M_{x^t i} = 0 \text{ but } M_{x^{t+1} i} = 1] \leq \Pr\left[\Delta + u(x^0, y^{(t)}) - u(x^1, y^{(t)}) \leq \alpha_1 \leq \Delta\right].$$

Based on the fact that we've drawn α from $\mathcal{U}(0, \sqrt{T})$ and that we must draw $\alpha_1 = \sqrt{T}$ for the above event to happen

$$\Pr\left[\Delta + u\left(x^0, y^{(t)}\right) - u\left(x^1, y^{(t)}\right) \leq \alpha_2 \leq \Delta\right] \leq \frac{1}{\sqrt{T}}.$$

We can the take the union bound over d columns and the possibility that x^t had a 1 and x^{t+1} had a 0 to show the desired regret bound $\mathcal{O}\left(\frac{d}{\sqrt{T}}\right)$.

2 Lower Bounds for No-Regret Algorithms

We'll now double back and describe a proof for our motivating result for introducing d-structure: without structural assumptions, no-regret algorithms are not feasible for exponential numbers of experts.

Theorem 2.1. *Any no-regret algorithm, even with an offline oracle, cannot guarantee a regret of* $\leq \frac{T}{16}$ *with runtime better than* $\mathcal{O}(\frac{\sqrt{N}}{\log^3(N)})$.

To prove this we will look at the set of functions on the unit cube. But first, some useful definitions:

Definition 2.2. $f : \{0, 1\}^n \to \mathbb{N}$ is **globally consistent** if any local maximum is a global maximum. Let $\Gamma(i)$ signify the neighbors of i. Then,

$$(\forall j \in \Gamma(i) \qquad f(j) \leq f(i)) \implies \forall j \in \{0, 1\}^n \qquad f(j) \leq f(i)).$$

Definition 2.3. The **Aldous problem** asks that, given query access, i.e., evaluation of $f(\cdot)$ on a given point in the domain, to an arbitrary globally consistent function f, design a randomized algorithm that determines whether $f^* = \max_i f(i)$ is odd or even with probability $\geq \frac{2}{3}$.

It is well-known that the Aldous problem requires many queries. That is:

Theorem 2.4. *Any algorithm that solves the Aldous Problem has to take* $\Omega\left(\frac{2^{\frac{d}{2}}}{d^2}\right)$ *queries to* f *in the worst case.*

We use Theorem 2.4 to show that even having access to an offline optimization, any no-regret algorithm has a runtime that is more than $\Omega\left(\sqrt{N}/\log^3(N)\right)$. At a high level, our proof would follow four steps

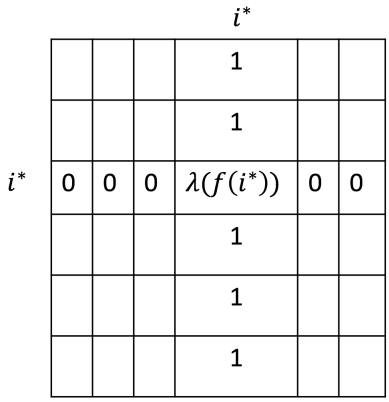


Figure 1: Game G defined below.

1. Given an f, construct a game where the minmax value is a if f^* is odd and is b if f^* is even. And $|a - b| \geq \Omega(1)$.

2. Make sure that the time spent on making a call to the best response oracle is comparable to the number of queries that we have to make to f to perform/implement the oracle.

3. Show that a no-regret algorithm pair with a best response oracle gives us a way to get arbitrarily close to the minmax value of the game. This we already did on the April 21 Lecture and you will practice another variant of it in Homework 5. This means that by playing a no-regret algorithm against itself, we can learn the minmax value of the game to accuracy $O\left(1/\sqrt{T}\right)$. And as a result a no-regret algorithm reveals whether f^* is odd or even.

4. So, the existence of a fast no-regret algorithm that uses an oracle implies the existence of an algorithm for solving Aldous Problem. Use step 2 to show that the number of queries would be small. By Definition 2.3, we know that this cannot be the case and, thus, there can be no fast no-regret algorithm as per Theorem 2.1

Let us now see some of the details of such a game that satisfied points 1 and 2. Let us interpret $k \in \mathbb{Z}$ as the analog of the nodes on the cube. Let

$$\lambda(k) := \begin{cases} \frac{1}{4} & k \text{ is even} \\ \frac{3}{4} & k \text{ is odd} \end{cases}$$

and define the game matrix to be

$$G = \begin{cases} \lambda(f(i)) & \text{if } i, j \text{ are both global maxima} \\ 0 & f(i) \geqslant f(j) \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

It is clear to see that $\lambda(f(i))$ is the minmax value. For a global maximum i^*, if the row player goes first their best action is e_{i^*}. The column player will also play e_{i^*}.

$$\min_P \max_Q G(P,Q) \leq \max_Q G\left(P^*,Q\right) = \lambda$$

If the column player goes first they will play e_{i^*} and then the row player will also play e_{i^*}.

$$\lambda = \min_P G\left(P,Q^*\right) \leqslant \max_Q \min_P G(P,Q)$$

Using a fast no-regret algorithm you can approximate the minmax value. Given the minmax value you can tell if $f*$ is is odd or even, solving the Alduos problem.

For point (2), we need to show that we can simulate the oracle calls using a small number of queries. For that, one can show that the best-response of the row player to Q, i.e., $BR(Q) := \text{argmin}_{j\in[N]} e_j^\top GQ$, is

$$BR(Q) = \text{argmax}_{i\in\Gamma(\text{Supp}(Q))} f(i).$$

We will leave this as an exercise.

Now note that each oracle call computes $\text{argmax}_{i\in\Gamma(\text{Supp}(Q))} f(i)$ which can be computed using $|\text{Supp}(Q)| \times \log_2(N)$ number of queries. That is because each point on the cube has $\log_2(N)$ neighbors. By Aldous's problem then $|\text{Supp}(Q)| \in \Omega\left(\sqrt{N}/\log_2^3(N)\right)$. But now note that the time it takes to run the online algorithm is at least the time needed to create the input for the oracle, which takes at least $|\text{Supp}(Q)| \in \Omega\left(\sqrt{N}/\log_2^3(N)\right)$.

References

Miroslav Dudik, Nika Haghtalab, Haipeng Luo, Robert E Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient online learning and auction design. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 528–539. IEEE, 2017.

Elad Hazan and Satyen Kale. Online submodular minimization. *Journal of Machine Learning Research*, 13(Oct):2903–2922, 2012.

Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 128–141, 2016.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.