CS6781 - Theoretical Foundations of Machine Learning

Lecture 21: Oracle-Efficient Online Learning I

April 23,2020

Lecturer: Nika Haghtalab *Readings: N/A*
Scribe: Jikun Wang and Tianjiao Li

In the previous lectures, we spent most of the time focusing on the statistical considerations in online learning, such as the Littlestone dimension, regret, partial vs. full information, etc. In this lecture, we focus on the computational aspects of online learning. Formally, considering the online learning framework where we have n experts, we want to know the runtimes over T timesteps as a function of n and T. To isolate the effects adversarial nature of *online* learning has on the runtime of these algorithms from the inherent difficulty of the task even in absence of an adversary, we will compare the runtime of *(adversarial) online learning* to the runtime of *(stochastic) offline learning*.

1 Runtime of Online Learning Algorithms

Let us recap the runtime of some of the online algorithms we studied in this course.

Randomized Weighted Majority In the Randomized Weighted Majority (RWM) algorithm, we performed the following update rule at each time step t,

$$w_i^{t+1} \leftarrow w_i^t(1-\epsilon)^{c_i^t} \text{ for all } i \in [n]$$

Later we draw i^{t+1} proportional to w_i^{t+1}. This results in a runtime of $O(nT)$ over T time steps.

Follow the Perturbed Leader In the Follow the Perturbed Leader (FTPL) algorithm, we first need to generate a random vector c^0, i.e., the regularizer or the cost function at step 0, and then we find the expert whose performance on time steps $0 \ldots, t-1$ was optimal. That is,

Step 0: Create a random cost $\vec{c}^0$, where $\vec{c}^0 \in \mathbb{R}^n$
Step t: Find expert $i^t = \underset{i}{\operatorname{argmin}} \sum_{\tau=0}^{t-1} c_i^t$, $i \in [n]$

As for the runtime, note that it tames $O(n)$ time to create an n-dimensional random vector $\vec{c}^0$. Naively implemented, at time step t we can observe the performance one expert over t time steps, in time $O(t)$ and choose the best one in total time $O(nT)$. Combining these together, the overall runtime of FTPL is $O(nT)$ as well.

Comparison to Offline Learning and ERM However, a key observation is that in many cases computing $i^t = \operatorname{argmin}_i \sum_{\tau=0}^{t} c_i^{t-1}$, $i \in [n]$ can be a lot faster, and may not require enumerating every expert and its performance. After all, this is precisely the type of computation ERM performs. For example, algorithms for linear optimization or algorithms for finding the shortest paths in a graph, search for the optimal options without enumerating the infinitely or exponentially large number of options.

As a concrete example, the online shortest path problem (think about the "driving to work" example in previous lectures).

- Experts are paths in the graph, denotes by indicator vectors in $|E|$ dimension: $\{0,1\}^{|E|}$
- Number of experts (paths): exponential in $|E|$
- Cost vectors: $\vec{c} \in [0,1]^{|E|}$ showing the cost per edge.

Let's consider the runtime of FTPL on this problem. For the generation of $\vec{c}^{\,0}$, the runtime is $|E|$. For solving the optimization problem at step t, we spend $O(T)$ time to aggregate the cost function $\vec{C}^{\,t} = \sum_{\tau=0}^{t-1} \vec{c}^{\tau}$ (or $O(1)$ if we do not recompute the past). Then we just need to find the shortest path with respect to costs in $\vec{C}^{\,t}$. Recall that Dijkstra's algorithm is known to find the shortest path in $O(|E| + |V| \log |V|)$ time. Finally the total runtime of FTPL runtime is much smaller than $O(Tn) = O(T \exp(|E|))$ that was claim above.

From this example we see that for problems where the number of experts is exponential in the natural representation of the problem the runtime $O(nT)$ can be prohibitively large. This includes many combinatorial optimization problems, games, and hypothesis classes. While it may not be clear how we can use RWM to take use of ERM, FTPL may be compatible with such use case. In the remainder of this lecture, we formally introduce the learning paradigm that allows online optimization algorithms such as FTPL to rely on offline optimization algorithms such as ERM.

2 Oracle-Efficient Online learning

2.1 Offline optimization oracle

In the language of this lecture, an offline optimization oracle is another way of calling empirical risk minimization (ERM). That is, OPT for a domain (expert class) $\mathcal{X}$ and set of cost functions $\mathcal{C}$, takes as input a set of cost functions $c^\tau \in \mathcal{C}$ for $\tau \in [t]$ and returns

$$\mathrm{OPT}_{\mathcal{X},\mathcal{C}}\left(\{c^\tau\}_{\tau=0}^{t}\right) = \operatorname*{argmin}_{i \in \mathcal{X}} \sum_{\tau=0}^{t} c^t(i).$$

When $\mathcal{X}$ and $\mathcal{C}$ are clear from the context, we suppress them in the above notation. Given the offline optimization oracle, we can call it to compute on any historical set of cost for the best expert. We seek to design oracle-efficient learners, that is, learners that run in polynomical time, with each oracle call counting $O(1)$ (Dudik et al. [2017]).

2.2 Oracle-Efficient Offline Learning

Theorem 2.1. *Offline optimization oracle is sufficient for efficient offline learning.*

This theorem basically says that, if the cost functions $c^t(\cdot) \sim D$, i.i.d over time, then playing ERM

$$i^t = \text{OPT}\left(\{c^\tau\}_{\tau=0}^{t-1}\right)$$

gets $err \leq O(\sqrt{T\log(n)})$ and runs in time T.

We pretty much proved this theorem before when we showed how one can learn in agnostic PAC setting using oracle. Recall that in PAC learning we have

$$m = \epsilon^{-2}\left(\log(|\mathcal{H}|) + \log(\frac{1}{\delta})\right)$$

Accordingly, we can see m as T, ϵ as average regret over T times, and $|\mathcal{H}|$ as n. Then we get the same error bound.

2.3 Oracle-Efficient Online Learning

The theory above involves no adversary, i.e., the costs $c^t(\cdot) \sim D$ are fully stochastic while the optimal i^t is deterministic. But for online learning, $c^t(\cdot)$ is picked adversarially, not randomly. Now a question arises: is offline optimization oracle sufficient for efficient online learning?

Theorem 2.2 ([Hazan and Koren, 2016])**.** *No algorithm, even with access to an offline optimization oracle, can guarantee a regret $O(T/2)$ using $\tilde{O}(\sqrt{n})$ runtime.*

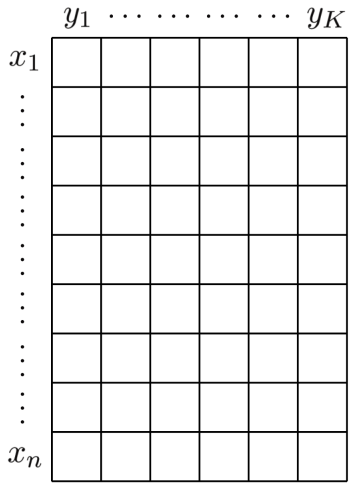
This theorem basically says, for general functions, online learning is strictly harder than offline learning. It is disappointing for us that for general functions we cannot hope for a decent regret bound in $\log(n)$ time. The good news is that, for many function classes that we typically use, there are oracle-efficient online algorithms. Before we specify what "typically" means, let's introduce the following structure:

For ease of representation, let $c^t(x^t) = u(x^t, y^t)$, where u is a predefined function known to the algorithm. In this game, x^t is our action and y^t is the action that the adversary takes. E.g., when the costs are linear we have $c^t(x^t)$ we define the loss as $u(x^t, y^t) = x^t \cdot y^t$ and use y^t is the vector corresponding the linear $c^t(\cdot)$.

The following theorem shows one can run polynomial in the number of actions of the adversary, instead of polynomial in terms of the actions of the learners (i.e., the number of experts).

Theorem 2.3. *For any loss function $u : \mathcal{X} \times \mathcal{Y}$, there is an oracle-efficient algorithm that runs in $poly(K, T)$ and gets regret bound of $O\left(K\sqrt{T}\right)$, where $K = |\mathcal{Y}|$ is the number of the actions available to the adversary.*

How can we remove the runtime dependence on the number of experts, i.e., $|\mathcal{X}|$? Let the following matrix be the game matrix encoding utilities $u(x_i, y_j)$.



ERM applies the oracle at each timestep t to everything we have observed from $\tau = 1$ to $t-1$ (the history):

$$\text{play } x^t \leftarrow \text{OPT}\left(\{y_\tau\}_{\tau=1}^{t-1}\right)$$

This is the ERM algorithm that has no dependence on the number of rows or columns, but it is not as we know it is not no-regret as it is a deterministic algorithm.

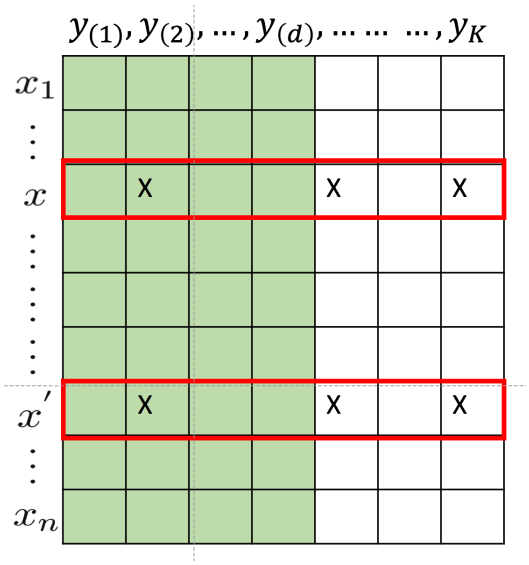
We want to introduced some randomness (in the style of FTPL) while still trying to pick the expert that was performing well historically? One way we could do this is altering the history of the play, by adding to the history some fake "hallucinated history". The particular form of "hallucinated history" that works well includes taking α_i copies of adversarial action $y_i \in \mathcal{Y}$ and adding it to the existing history, where $\alpha_i \sim D$ are appropriately defined random variables.

$$\text{Play } x^t \leftarrow \text{OPT}\left(\{y^\tau\}_{\tau=1}^{t-1} \cup \bigcup_{i=1}^{K} \alpha_i y_i\right),\ \forall i \in [K], \alpha_i \sim D$$

This algorithm tuns in time $\text{poly}(K, T)$ as we need to add each of the K adversarial actions to the history and is indeed the algorithm that gets the guarantees of Theorem 2.3 (when used with a specific D). Unfortunately, in many cases the set of adversarial actions is also very large or even infinitely large, which makes the runtime and regret of Theorem 2.3 less than desirable. We show how direct dependence on K in unnecessary when the problem has some additional structure.

Definition 2.4. 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 2.5. *If the problem has a d-structure and the utilities are 0-1, then you can have a no-regret algorithm with regret $d\sqrt{T}$ and oracle-efficient runtime of $poly(d, T)$.*[1]

This theorem is identical to Theorem 2.3 except for that we replace K with d. Let's demonstrate with an example a case where Theorem 2.3 would lead to an exponential runtime, but Theorem 2.4 gives a polynomial runtime. Recall from the online shortest path setting that

- rows: experts $\in \{0, 1\}^{|E|}$
- columns: each column is one way of putting cost on all edges. If the cost is $\{0, 1\}$ discrete cost, there are $2^{|E|}$ columns.
- cost: assume linear cost.

In this problem, $d = |E|$. That is let $y_{(j)}$ refer to the cost function that assigns a cost of 1 to edge $j \in E$ and cost 0 to all other edges. Then for any two different paths x and x' there is an edge j that appears in one of them only. In that case $u(x, y_{(j)}) \neq u(x', y_{(j)})$. So, the set of $\{y_{(j)}\}_{j \in E}$ forms a d-structure.

2.4 Warming up for next time

Next time we will analyze *Generalized FTPL* for oracle efficient learning when the problem has a d-structure. 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

[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.

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**

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 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.