CS6781 - Theoretical Foundations of Machine Learning

Lecture 11: Introduction to Online Learning

February 27, 2020

Lecturer: Nika Haghtalab *Readings: Chapter 21-21.2, UML*
Scribe: Katherine Van Koevering

1 Overview

So far, we have talked about the PAC model where samples are drawn i.i.d. from distribution $\mathcal{D}$ and the goal was to find some $h \in \mathcal{C}$ with low error on distribution $\mathcal{D}$. Next, we will talk about a learning model where we make no assumptions about how the data is generated – we do not assume that there is a distribution that generates the data. Without this assumption, what is the right performance benchmark? And how do we achieve it? This is what we talk about today.

2 Online Model

Consider a scenario where we make some decision every day with no information on what today is going to look like ahead of time, but we do know what decisions we have made in the past and how they performed. An example is deciding which route to take to drive to Cornell everyday without knowing After driving, we can look back and asses the traffic on our route and perhaps also ask others what the traffic was like on all the other routes in Ithaca. What you want is that, over a long period of time, we don't do much worse than if we took the single best route in hindsight.

In another example, consider the setting where everyday we guess whether the stock market index is going to go up or down. You may rely on some recommendation sources: Wall Street Journal, CNN, your neighbor, your psychic, etc. At the end of the day, you observe whether you were right or wrong. Your goal is not to make many more mistakes compared to your best advisor in hindsight.

2.1 Mistake Bound Model

We start formalizing this setting via the Mistake Bound model. Given a class of hypothesis $\mathcal{C}$, for $t = 1, 2, 3, ...$

- Receive $x^t \in \mathcal{X}$.
- Predict $\hat{y}^t \in \{-, +\}$.
- Receive true label $y^t = c^*(x^t)$ for some $c^* \in \mathcal{C}$. A mistake is made if $\hat{y}^t \neq y^t$

The goal is to bound the number of mistakes you make. Note that we have made a realizability assumption, that is there is an unknown $c^* \in \mathcal{C}$ such that for all t, $y^t = c^*(x^t)$. At every timestep t, the algorithm has access to the sequence of past experience, i.e., $\{(x^1, y^1), (x^2, y^2), \cdots, (x^{t-1}, y^{t-1})\}$.

Definition 2.1. An algorithm $\mathcal{A}$ learns the hypothesis class $\mathcal{C}$ with mistake bound M if for any online sequence $(x^1, y^1), (x^2, y^2), ...$ for which there is $c^* \in \mathcal{C}$, such that $y^t = c^*(x^t)$ for all t, the algorithm makes at most M mistakes.

Ideally, you want both M and the runtime of the per-step runtime of the algorithm to be polynomial in terms of the instance size.

Example 2.2. *Let $\mathcal{X} = \{0,1\}^n$ and $\mathcal{C}$ be the set of monotone disjunctions. Can you find an algorithm with per-round runtime of $poly(n)$ and mistake bound of $M = n$?*

Note that the realizability assumption means that there is a $P^* \subseteq [n]$ for which $c^*(x) = \bigvee_{i \in P^*}(x_i)$. Initialize $P = [n]$ and define $h(x) = \bigvee_{i \in P}(x_i)$. For timestep t, use h to predict $\hat{y}^t = h(x^t)$. If we make a mistake on x^t, then update $P \leftarrow P \setminus \{i : x_i^t = 1\}$. Note that we will never predict $h(x^t) = -$ when $c^*(x) = +$, because $P^* \subseteq P^t$. Also, whenever we make a mistake the size of P^t decreases by at least 1, so we make no more than n mistakes. Furthermore, this is tight. If $P^* = \emptyset$ and all the vectors are unit vectors, then $M = n$. Run time is $O(n)$ per step, since we remove at most n elements.

2.2 Majority/Halving Algorithm

For the remainder of the lecture, let's forget for a moment about computational efficiency, and focus on having a low mistake bound. We are still dealing with a class $\mathcal{C} = \{h_1, \cdots, h_n\}$ of hypotheses. It's common to refer to these hypotheses and their predictions as "expert's advice". Each expert will make a prediction at every time step t, and you can observe the history of predictions by every expert to make a decision at time t. In this section, we will continue making the realizability assumption: there is some expert h^* that is always right. For now we will deal with binary classification - either we are right or we are wrong.

Majority/Halving Algorithm Algorithm: At time t, consult all h_i that have yet to make a mistake. Go with their majority vote.

Theorem 2.3. *The Majority Algorithm has mistake bound $M \leq \lfloor \log_2 |\mathcal{C}| \rfloor$.*

Proof. Note that the mistake bound model assumes that there is some consistent $c^* \in \mathcal{C}$ such that $\forall t, c^*(x^t) = y^t$. Let $S_t = \{h_i : h_i \text{ has not made a mistake so far}\}$ and let $W_t = |S_t|$. If the majority make a mistake, then at least half of S_t were wrong. Thus, $W_{t+1} \leq 0.5W_t$. Furthermore, we know $W_t \geq 1$ for all t since we assume there exists a "perfect" expert that is always right. This implies that you can halve W_t at most $\lfloor \log_2(|\mathcal{C}|) \rfloor$ times. □

2.3 Weighted Majority

Notice that the Majority algorithm relies on the existence of a perfect expert, i.e., a consistent hypothesis. What if there is no consistent $c^* \in \mathcal{C}$? Can we have a mistake bound M that is not too much worse than the best possible $c \in \mathcal{C}$?

Given a sequence $\{(x^1, y^1), \dots\}$, let $\mathbf{OPT}$ be the number of mistakes the best expert makes. We would like to have an algorithm that makes a number of mistakes, M, that is not much worst than $\mathbf{OPT}$. That is, for any sequence of inputs, $M \leq \alpha\mathbf{OPT} + \beta$ for desirable values of α and β.

One possible strategy is the Iterated Majority Algorithm, where we run the Majority Algorithm until we run out of experts, reinstate all the experts, and start again. Note that for this algorithm, $M \leq \log|\mathcal{C}|(\mathbf{OPT} + 1)$. This is because, for every mistake the best expert makes the iterated Majority Algorithm algorithm can take up to $\log|\mathcal{C}|$ mistakes and run out of the experts, and reinstate all experts again.

Iterated Majority Algorithm has a large mistake bound compared to $\mathbf{OPT}$. This is because everytime the algorithm restarts, it forgets the performance of the experts so far. In order to improve this algorithm, instead of eliminating experts that make a mistake, we can reduce their credibility. We can then take this credibility into account when we make our decision. This motivates the following algorithm:

Algorithm 1 Weighted Majority (ϵ)

```
Let $w_1^1, w_2^1, ..., w_n^1 = 1$.
for $t = 1, 2, 3, ...$ do
    if $\sum_{i : h_i(x^t)=1} w_i^t > \sum_{i : h_i(x^t)=0} w_i^t$ predict 1. Otherwise predict 0
    Receive $y^t$
    Define $E_t = \{i : h_i(x^t) \neq y^t\}$
    For each $i \in E_t$, update $w_i^{t+1} = (1 - \epsilon)w_i^t$
end for
```

Theorem 2.4. *For any sequence of inputs and any time $T > 0$, let M be the number of mistakes the Weighted Majority algorithm makes, and $\mathbf{OPT}$ be the number of mistakes made by the best*

$h \in \mathcal{C}$. Then,

$$M \leq \frac{2}{1-\epsilon}\mathbf{OPT} + \frac{2}{\epsilon}\ln|\mathcal{C}|.$$

Instead of proving the general case, we will prove that for $\epsilon = 0.5$, $M \leq 2.4\mathbf{OPT} + \log|\mathcal{C}|$.

As we did for the analysis of the Majority Algorithm, we use W_t to roughly represent the amount of credibility we have left in the set of our experts. Every time we make a mistake, this credibility is reduced. This gives us an upper bound on the total credibility after having made some number of mistakes. We use these two bounds to show that our algorithm does not make too many mistakes compared to the best expert.

Proof. At some time T, $W^T = \sum_{i=1}^n w_i^T$. Let i^* be the best expert, which makes at most $\mathbf{OPT}$ mistakes, and thus has their weight halved $\mathbf{OPT}$ times. It follows that the sum of the weights after the algorithm has ended, is at least the weight of the best expert, i.e.,

$$W^{T+1} \geq w_{i^*}^{T+1} = \left(\frac{1}{2}\right)^{\mathbf{OPT}}.$$

Now we upper bound the remaining credibility — that is, showing that when we make a mistake the total weight reduces significantly. When Weighted Majority is wrong, then the weight of the correct experts formed a minority. Recall we denoted the set of wrong experts as E_t. So whenever we make a mistake at time t, we have

$$\sum_{i \notin E_t} w_i^t \leq \frac{W^t}{2}$$

Thus,

$$\begin{aligned} W^{t+1} &= \frac{1}{2}\sum_{i \in E_t} w_i^t + \sum_{i \notin E_t} w_i^t \\ &= \frac{1}{2}(W^t - \sum_{i \notin E_t} w_i^t) + \sum_{i \notin E_t} w_i^t \\ &= \frac{1}{2}W^t + \frac{1}{2}\sum_{i \notin E_t} w_i^t \\ &\leq \frac{3}{4}W^t. \end{aligned}$$

Note that $W^1 = |\mathcal{C}|$. How many times can you multiply W^1 by $\frac{3}{4}$ stay above $\frac{1}{2}^{\mathbf{OPT}}$?

$$\frac{1}{2}^{\mathbf{OPT}} \leq W^{T+1} \leq \frac{3}{4}^{M}|\mathcal{C}|$$

$$\implies M \log \frac{4}{3} \leq \log |\mathcal{C}| + \mathbf{OPT}$$
$$\implies M \leq 2.4(\mathbf{OPT} + \log |\mathcal{C}|)$$

□

Consider the general form of the above theorem, $M \leq \frac{2}{1-\epsilon}\mathbf{OPT} + \frac{2}{\epsilon} \ln |\mathcal{C}| \approx 2(1+\epsilon)\mathbf{OPT} + \frac{2}{\epsilon} \ln |\mathcal{C}|$. Note that, this bound essentially states that

$$M \leq 2\mathbf{OPT} + 2\epsilon\mathbf{OPT} + \frac{2}{\epsilon} \ln |\mathcal{C}| \leq 2\mathbf{OPT} + O\left(\sqrt{\mathbf{OPT} \ln(|\mathcal{C}|)}\right),$$

for the optimal parameter setting of $\epsilon = \sqrt{\frac{\ln(|\mathcal{C}|)}{\mathbf{OPT}}}$.

This is not a good bound for large values of $\mathbf{OPT}$. For instance, for $\mathbf{OPT} = 0.3T$, $M < 0.6T$ does not say much — we can ensure that $M = 0.5T$ in expectation with just random guessing. In the next section, we show how one can get a bound of $M \leq \mathbf{OPT} + O\left(\sqrt{\mathbf{OPT} \ln(|\mathcal{C}|)}\right)$. We show that one of the weaknesses of Weighted Majority algorithm is that it is a deterministic algorithm. Therefore, it does not perform well in presence of adversaries who choose a bad sequence specifically to make our algorithm have a bad mistake bound (See lecture 13 to see why deterministic algorithms have bad performance). Therefore, to improve the multiplicative dependence from $2\mathbf{OPT}$ to $\mathbf{OPT}$, we need to use randomness.

2.4 Randomized Weighted Majority

In this section, we extend the setting to arbitrary costs, instead of focusing on $0/1$ error. Formally, we consider the following setting. We again have n hypotheses or experts, e.g., representing different routes we take to work or different people we ask about the stock market, indexed by $[n]$. At every time step t, an adversary designs a cost function $c^t : [n] \to [0, 1]$, e.g., representing traffic or time delay, having full knowledge of our algorithm and the history. The algorithm chooses an expert i^t to follow and pays cost $c^t(i^t)$. The algorithm sees the cost of *every* expert afterwards. Suppose we play for T rounds. We want to compare the cost our algorithm pays, i.e., $\sum_{t=1}^{T} c^t(i^t)$, to the cost of the best expert, i.e., $\min_{i\in[N]} \sum_{t=1}^{T} c^t(i)$. A randomized version of the weighted majority algorithm is as follows:

Theorem 2.5. *At any time $T > 0$, with an adaptive adversary, for Randomized Weighted Majority,*

$$\mathbb{E}\left(\sum_{t=1}^{T} c^t(i^t)\right) \leq \frac{1}{1-\epsilon}\mathbb{E}\left(\min_{i\in[n]} \sum_{t=1}^{T} c^t(i)\right) + \frac{1}{\epsilon}\ln(n),$$

where the expectation is taken over the randomness of the algorithm and adversary.

Algorithm 2 Randomized Weighted Majority (ϵ)

```
$w_1^1, w_2^1, ..., w_n^1 = 1$
for $t = 1, 2, 3, ...$ do
    $W^t = \sum_{i=1}^{n} w_i^t$
    For $i \in [n],\ p_i^t = \frac{w_i^t}{W^t}$
    Sample $i^t$ from distribution given by $p^t = (p_1^t, \ldots, p_n^t)$.
    Receive the cost function $c^t$ and pay the appropriate cost $c^t(i^t)$.
    For all $i : w_i^{t+1} \leftarrow w_i^t \cdot (1 - \epsilon)^{c^t(i)}$
end for
```

Note that Theorem 2.5 improves on Theorem 2.4 by a factor of 2. Note that when $c^t(i)$ is the $0/1$ cost function for binary mistakes, Theorem 2.5 implies that $\mathbb{E}[M - \mathbf{OPT}] \lessapprox \epsilon\mathbf{OPT} + \frac{1}{\epsilon}\ln(n) \in O\left(\sqrt{\mathbf{OPT}\ln(n)}\right)$ for the optimal parameter setting of $\epsilon = \sqrt{\ln(n)/\mathbf{OPT}}$. Therefore, this is a big improvement over the type of results the deterministic Weighted Majority algorithms achieves in Theorem 2.4. We will prove this theorem in the next lecture.