CIS5200: Machine Learning Spring 2023

Lecture 23: Online Learning

Date: April 11, 2023 *Author: Surbhi Goel*

Acknowledgements. These notes are heavily inspired by notes by Nika Haghtalab (UC Berkeley) and Sham Kakade (Harvard).

Disclaimer. These notes have not been subjected to the usual scrutiny reserved for formal publications. If you notice any typos or errors, please reach out to the author.

1 Online Learning

So far we have looked at setting where we were given a training dataset (labelled or unlabelled) drawn i.i.d. from some underlying distribution $\mathcal{D}$, and our task was to learn either a useful prediction function or underlying structure of the data. However, in many settings, such as spam classification, we need to make decisions in a sequential manner (per email) while being adaptive to the adversary (spammer) reacting to our prediction (spam classifier). In today's class we will formalize the model of online learning, discuss how to measure performance, and look at a few algorithms that can successfully learn a good predictor in this model of learning.

1.1 Setup

We will focus on binary classification over domain $\mathcal{X}$, as we did for PAC learning. Online learning will be performed in a sequence of T consecutive rounds, where for each round $t = 1, \ldots, T$:

1. Learner is given an instance $x_t \in \mathcal{X}$ either from the environment or an adversary
2. Learner makes a prediction $\hat{y}_t \in \{-1, 1\}$
3. Learner observes actual label $y_t \in \{-1, 1\}$
4. Learner suffers a loss $\ell(\hat{y}_t, y_t)$ (*we will assume 0-1 loss for this lecture*)

The goal of the learner is to ideally minimize the total loss it suffers over the T rounds. The only way for this to be possible is if the learner can deduce some information about the future inputs given the past. So far, we used the i.i.d. assumption (data is drawn i.i.d. from an unknown distribution $\mathcal{D}$) on the input datapoints to quantify this. However, in online learning we make no such assumption (*worst case setup*) in general, we allow the input sequence to be either deterministic, stochastic, or even adversarially adaptive to the learner. In this setting, the adversary can make us make mistakes on every round by just choosing $y_t = -\hat{y}_t$. Therefore, we need to make additional assumptions.

2 Mistake Bound Model

One possible assumption is to assume *realizability* (like we did in PAC learning), that is, for all t, $y_t = f(x_t)$ for some underlying $f \in \mathcal{F}$. Now we can hope that the learner makes only few mistakes for any T even if f and $x_1, \ldots, x_T$ are chosen by the adversary. The worst-case number of mistakes any learner $\mathcal{L}$ makes is known as the mistake bound of $\mathcal{L}$ for function class $\mathcal{F}$. More formally,

$$M_{\mathcal{L}}(\mathcal{F}) := \max_{\substack{f \in \mathcal{F}, T \\ x_1, \ldots, x_T \in \mathcal{X}}} \sum_{t=1}^{T} \mathbb{1}[f(x_t) \neq \hat{y}_t],$$

where $\hat{y}_t$ are the predictions of $\mathcal{L}$ on the sequence.

Definition 1 (Online learnability, realizable)**.** *We say that a learner $\mathcal{L}$ online learns function class $\mathcal{F}$ with mistake bound B iff*

$$M_{\mathcal{L}}(\mathcal{F}) \leq B < \infty.$$

Here we ignore computational efficiency, and only focus on minimizing the mistake bound.

Let us assume that the function class $\mathcal{F}$ is finite. Then we can describe a simple learner that has a finite mistake bound:

Algorithm 1: Halving

Initialize $\mathcal{V}_1 = \mathcal{F}$
for $t = 1, 2, \ldots$ **do**
  Receive x_t
  Predict $\hat{y}_t = \arg\max_{y \in \{-1,1\}} |\{f \in \mathcal{V}_t : f(x_t) = y\}|$ (in case of tie, predict $\hat{y}_t = 1$)
  Receive true label y_t
  Update $\mathcal{V}_{t+1} = \{f \in \mathcal{V}_t : f(x_t) = y_t\}$
end

Theorem 2. *Let $\mathcal{F}$ be a finite hypothesis class. The Halving algorithm enjoys the mistake bound $M_{Halving}(\mathcal{F}) \leq \log(|\mathcal{F}|)$.*

Proof. Observe that whenever the algorithm makes an error we have $|\mathcal{V}_{t+1}| \leq \frac{|\mathcal{V}_t|}{2}$ (hence the name Halving) since the algorithm predicts according to the majority. Therefore, if M is the total number of mistakes, we have

$$1 \leq |\mathcal{V}_{t+1}| \leq \frac{|\mathcal{F}|}{2^M} \implies M \leq \log(|\mathcal{F}|).$$

□

Recall that in offline learning, all ERMs got the same guarantee (PAC results). However, in this model, you don't get that. Consider the following learner:

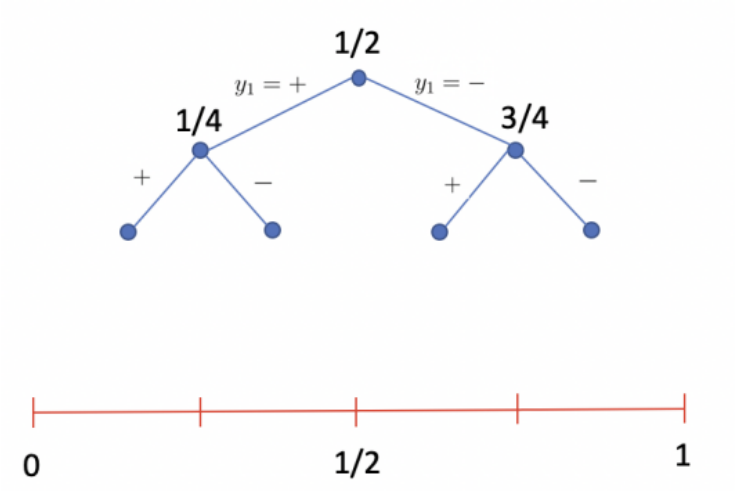

Figure 1: Adversarial sequence for thresholds. Source: Nika Haghtalab's lecture notes https://www.cs.cornell.edu/courses/cs6781/2020sp/lectures/12_online2.pdf.

Algorithm 2: Consistent

Initialize $\mathcal{V}_1 = \mathcal{F}$
for $t = 1, 2, \ldots$ **do**
  Receive x_t
  Choose any $f_t \in \mathcal{V}_t$
  Predict $\hat{y}_t = f_t(x_t)$
  Receive true label y_t
  Update $\mathcal{V}_{t+1} = \mathcal{V}_t \setminus \{f_t\}$
end

It is not too hard to see that the mistake bound would be $|\mathcal{F}| - 1$. In fact, you can construct a setting in which this is achieved. Therefore, different ERMs have different performance.

Example: VC Dimension is not enough. Consider the class of linear thresholds (that we have looked at several times before) defined by a parameter a, $f_a(x) = \begin{cases} 1 & \text{if } x \geq a \\ -1 & \text{otherwise.} \end{cases}$. Recall that the VC dimension of this class is 1 so $O(1/\epsilon)$ samples are enough to learn this class in the realizable offline setting. In the adversarial online setting, we can show a strategy of the adversary that guarantees infinite mistakes. The idea is similar to the active learning strategy to improve sample complexity, we give the adversary point 0 and 1 with labels -1 and 1 respectively. Then we choose the point $1/2$, and give it a random label. Depending on this label, we go to the corresponding half and repeat this. See Figure 1. This would lead to $T/2$ mistakes in expectation.

Note: Similar to VC dimension, there is a notion of Littlestone dimension that characterizes online learnability for infinite function classes. This is out of the scope of this course.

Example: Perceptron The perceptron algorithm has an online version as follows:

Algorithm 3: Perceptron

Initialize $w_1 = 0$
for $t = 1, 2, \ldots$ **do**
  Receive x_t
  Predict $\hat{y}_t = \text{sign}\left(w_t^\top x_t\right)$
  Receive true label y_t
  if $\hat{y}_t \neq y_t$ **then** Update $w_{t+1} = w_t + y_t x_t$
  else Update $w_{t+1} = w_t$
end

You can perform a very similar analysis to the batch perceptron version we looked at before and show that mistake bound is $1/\gamma^2$ under the assumption that $\|w_*\| = 1$ and for all t, $\|x_t\| \leq 1$.

3 Regret

If we do not make the assumption of realizability, it is not always possible to have a mistake bound. Similar to agnostic PAC learning, here we use a notion of *regret*, which captures how much the learner could have improved if it chose a fixed function from the function class in retrospect. More formally, the regret of a leaner $\mathcal{L}$ relative to a function class $\mathcal{F}$ on the input sequence is

$$\text{Regret}_{\mathcal{L}}(\mathcal{F}, T) = \sum_{t=1}^{T} \ell(\hat{y}_t, y_t) - \min_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell(f(x_t), y_t).$$

Now the updated goal is to minimize regret, that is, do as well as any function in the function class could do if it had access to the entire sequence (as in the offline supervised setting). Not that when we say minimize regret, our goal is to get the regret to be sublinear in T (think $\log T$ or $\sqrt{T}$) so that as T tends to infinity, the average regret ($\text{Regret}_{\mathcal{L}}(\mathcal{F}, T)/T$) tends to 0.

Definition 3 (Online learnability, general)**.** *We say that a learner $\mathcal{L}$ online learns function class $\mathcal{F}$ if for any sequence,*

$$\lim_{T \to \infty} \frac{\textit{Regret}_{\mathcal{L}}(\mathcal{F}, T)}{T} = 0.$$

Such a leaner is also called a no-regret learner.

4 Weighted Majority

Working with finite class $\mathcal{F}$, we can generalize the halving algorithm for the non-realizable setting. Let us view each function $f \in \mathcal{F}$ as an *expert* who is assisting with our prediction. Similar to boosting, we can use a weighted vote of these experts to decide how to predict, and subsequently update the weights to reflect how good we think each expert is. Here's the algorithm:

Algorithm 4: Weighted Majority

Initialize $w_{1,i} = 1$ for all $i \in [n]$
for $t = 1, 2, \ldots$ **do**
  Receive x_t
  Predict $\hat{y}_t = \text{sign}\left(\sum_{i=1}^{n} w_{t,i} f_i(x_t)\right)$
  Receive true label y_t
  Define $E_t = \{i : f_i(x_t) \neq y_t\}$ (set of all incorrect experts)
  if $i \in E_t$ **then** Update $w_{t+1,i} = w_{t,i}/2$
  else Update $w_{t+1,i} = w_{t,i}$
end

For this algorithm, we can bound the mistakes it makes on the sequence and the optimal number of mistakes.

Theorem 4. *Let $\mathcal{F}$ be a finite hypothesis class. Let M be the total number of mistakes made by the Weighted Majority algorithm, and let M^* be the number of mistakes made by the best expert, then*

$$M \leq 2.41(M^* + \log |\mathcal{F}|).$$

Proof. We denote the size of the set by $n = |\mathcal{F}|$. At each time step t, each expert makes a prediction, and we assign a weight $w_{t,i}$ to the i-th expert's prediction. We define the total weight at time step t as $W_t = \sum_{i=1}^{n} w_{t,i}$, and let i^* be the index of the best expert, who makes the fewest mistakes throughout the entire process. Note that when we make a mistake, the weight on the correct experts must have been less than half the total weight, that is,

$$\sum_{i \notin E_t} w_{t,i} \leq \frac{W_T}{2}.$$

We also have,

$$\begin{aligned}
W_{t+1} &= \sum_{i \in E_t} \frac{w_{t,i}}{2} + \sum_{i \notin E_t} w_{t,i} \\
&\leq \left(\frac{W_t}{2} - \sum_{i \notin E_t} \frac{w_{t,i}}{2} \right) + \sum_{i \notin E_t} w_{t,i} \\
&= \frac{W_t}{2} + \frac{1}{2} \sum_{i \notin E_t} w_{t,i} \\
&\leq \frac{3}{4} W_t.
\end{aligned}$$

Thus we have,

$$W_{T+1} \leq \left(\frac{3}{4}\right)^M n.$$

Now observe that function i^* makes M^* mistakes, therefore

$$W_{T+1} \geq w_{T+1,i^*} = \frac{1}{2^{M^*}}.$$

Combining the above, we get

$$M \leq 2.41(M^* + \log n).$$

□

Question: What happens when we add more good/bad experts (or increase our function class)?

There is an improved randomized version of this algorithm that gets the regret of $O(\sqrt{T \log |\mathcal{F}|})$. We will not cover this in class.