Indian Institute of Technology Kanpur
CS773 Online Learning and Optimization

Scribe: Viveka Kulharia
Instructor: Purushottam Kar
Date: January 12, 2016

LECTURE

4

Online Parameter Estimation

1 Introduction

With this lecture, we will commence with algorithms for online learning. We will look at the problem of predicting the outcomes of tosses of a fair coin, and study a natural algorithm for doing the same. This shall introduce us to a popular algorithmic paradigm called `Follow-the-leader`.

Although the problem of predicting the outcome of a coin toss may seem futile and without application, it actually underlies the basic process of estimating the parameters of a system in an online fashion.

1. Several systems we deal with are stochastic. For example, stock prices, interest rates and currency prices are stochastic and predicting the behavior of these systems frequently requires estimating latent parameters in an online fashion.

2. Parameter estimation is, in some sense, a precursor to learning. Intuitions gained in studying online parameter estimation would aid in motivating online learning algorithms.

2 Online Parameter Estimation

In this lecture we will work with what is arguably the simplest stochastic system, involving a single coin that is tossed repeatedly. Thus, at the heart of our system is a coin C with *bias* $p \in [0, 1]$. The bias of a coin is defined to the be the probability of it landing heads, i.e.

$$p := \mathbb{P}[C = H].$$

As discussed in Lecture 1, online learning involves a game between the learner and an adversary. In this system, the adversary would be tossing the coin and learner would try to predict the outcomes of the tosses. We will now formalize the setting and set the notation.

2.1 The Online Bit Prediction Process

Algorithm 1 gives a description of the online process (often, one refers to the online process as an *online* game instead). At each time step, the learner makes a *prediction* $\hat{y}^t$ of what it thinks would be the outcome of the next coin toss[1]. Next, the coin is actually tossed (by the adversary), and its outcome is revealed to the learner as the *feedback* y^t. Recall that the adversary is honest, that is to say, it correctly reveals the outcome of the coin toss. Depending on the prediction and the feedback, the learner incurs an *instantaneous loss* ℓ^t.

[1] We will use a random variable to encode heads by 1 and tails using 0.

Algorithm 1: Online Bit Prediction

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict $\hat{y}^t \in \{0, 1\}$
3: Receive $y^t \in \{0, 1\}$
4: Incur loss $\ell^t := \ell\left(\hat{y}^t, y^t\right)$
5: **end for**

This game can go indefinitely or terminate after a finite number of rounds or iterations. If it does terminate after a finite number T of rounds, then the number T is often called the *horizon* of the online process. In the future, we will be looking at online algorithms that are *horizon-aware*, as well as those that are *horizon-oblivious*. Horizon-aware algorithms will require the knowledge of how many rounds the online process will go on for, often in order to set hyperparameters to optimal values. Horizon-oblivious algorithms make no such assumption.

Remark 4.1 (From horizon awareness to pure bliss)**.** It is often possible to take a horizon-aware algorithm and convert to a horizon-oblivious one by using the so-called *doubling trick*. It involves making a guess T_0 about the horizon and taking all decisions (including hyperparameter settings) according to the guess. If the guess proves to be wrong and the online process continues beyond T_0 rounds then we update the guess to a new value, typically $T_1 = 2 \cdot T_0$ and proceed thereafter according to the new guess.

2.2 Instantaneous Loss

As mentioned before, the learner incurs a penalty or a loss at each time step. The loss, in this case, would be a function of the learner's prediction $\hat{y}^t$ and the true outcome y^t. The most general form of describing this instantaneous loss function is via a *loss matrix* which can be used to assign target specific losses etc. An example of a generic loss matrix is given in Table 1.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	a	b
$y = 1$	c	d

Table 1: A general loss matrix

For instance, we can have $b \gg c$ to indicate a higher penalty for predicting a tails as a heads than the other way round. Usually one has $a = c = 0$ to avoid penalizing the learner if the prediction is indeed correct. However, for this lecture, we will focus on the *misclassification/mislabelling loss* function which is a *symmetric* loss function that simply counts if there was a misprediction of the toss.

$$\ell^{0\text{-}1}\left(\hat{y}, y\right) := \mathbb{I}\left\{\hat{y} \neq y\right\}$$

The loss matrix corresponding to the misclassification loss is given in Table 2.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	0	1
$y = 1$	1	0

Table 2: The loss matrix corresponding to the 0-1 loss function.

2.3 Putting it all together

To complete our description of the online prediction process, we introduce the notion of *stream history*. At any given time t, the stream history contains all the information about the online process till that time instant i.e.

$$\mathcal{H}^t = \{(\hat{y}^\tau, y^\tau)\}_{\tau=1}^{t-1}$$

Any algorithm $\mathcal{A}$ performing prediction in this online game must utilize the stream history (and possibly some additional *advice* E^t) to decide its prediction.

$$\mathcal{A} : (\mathcal{H}^t, E^t) \mapsto \hat{y}^t$$

The advice may include randomness that randomized algorithms use to make stochastic predictions (we will see one such example in the next lecture) or else *expert advice* (yet again, examples will be provided in the next lecture). However, the algorithm is prohibited from peeking into the future, i.e. the algorithm does not have (even partial) access to the feedback or the instantaneous loss at time t or beyond. The sources of advice (experts etc) the algorithm uses are also not allowed to use such future information to constitute the advice.

Definition 4.1 (Cumulative Loss)**.** Suppose we execute the online process for T rounds, and let $\mathbf{y} \in \{0,1\}^T$ denote the vector of feedback received in these rounds. Then the cumulative loss of an algorithm $\mathcal{A}$ over the T rounds, using ℓ as the instantaneous loss function, is defined as

$$L^\ell(\mathcal{A}, \mathbf{y}) = \sum_{t=1}^{T} \ell\left(\mathcal{A}\left(\mathcal{H}^t, E^t\right), y^t\right).$$

For notation simplicity, we will often omit mention of the history and the advice and treat them (and the restrictions thereon) as implicit, and use the notation $\mathcal{A}^t := \mathcal{A}\left(\mathcal{H}^t, E^t\right)$. In our specific setting, where we are using the misclassification loss, we define the cumulative mistake as follows

Definition 4.2 (Cumulative Mistake)**.** Let the problem setting be as before and let the misclassification loss $\ell = \ell^{0\text{-}1}$ be used to decide the instantaneous penalties. Then the cumulative mistake incurred by an algorithm over T rounds is defined as

$$L^{0\text{-}1}(\mathcal{A}, \mathbf{y}) = \sum_{t=1}^{T} \ell^{0\text{-}1}\left(\mathcal{A}^t, y^t\right).$$

In the following, we shall assume that the feedback is decided by tossing the coin independently. Thus, y^t is sampled from *Bernoulli distribution* with bias p, i.e.

$$\mathbb{P}\left[y^t = 1 \mid \left\{\mathcal{H}^t \cup \hat{y}^t\right\}\right] = p,$$

since the toss is independent of the stream history and the current prediction.

3 A Worthy Benchmark

As discussed in Lecture 1, bounds on the cumulative mistake or cumulative loss are in themselves not very indicative of the performance of an algorithm. The cumulative mistake could be very large because the algorithm performed miserably. However, the cumulative mistake could also be large because the prediction problem was very hard. We need a way to distinguish between these two scenarios. Benchmarks help us out in this respect.

In our case the second point is especially relevant since our adversary is extremely unpredictable. Thus, it is essential to compare our performance to a reasonable *benchmark*. In the following discussion, we shall establish one such benchmark. Later, we will develop algorithms that can compete with this benchmark.

Recall that our algorithm uses history and additional advice to make the prediction. Without loss of generality, we can assume that the algorithm makes its prediction by tossing a coin with some bias $\hat{p}^t$ i.e.

$$\mathbb{P}\left[\hat{y}^t = 1 \mid \left\{\mathcal{H}^t, E^t\right\}\right] = \hat{p}^t$$

Note that in this case, the total advice to the algorithm would include E^t, as well as the random bits used to implement the above coin toss. Given this, we can calculate the expected instantaneous loss at this time step as follows (in the following, the conditioning on the history is implicit and omitted for the sake of notational simplicity):

$$\begin{aligned}
\mathbb{E}\left[\ell^{0\text{-}1}\left(\hat{y}^t, y^t\right)\right] &= \mathbb{E}\left[\mathbb{I}\left\{\hat{y}^t \neq y^t\right\}\right] \\
&= \mathbb{P}\left[\hat{y}^t \neq y^t\right] \\
&= \mathbb{P}\left[\left(\hat{y}^t = 1, y^t = 0\right) \cup \left(\hat{y}^t = 0, y^t = 1\right)\right] \\
&= \mathbb{P}\left[\hat{y}^t = 1, y^t = 0\right] + \mathbb{P}\left[\hat{y}^t = 0, y^t = 1\right] \\
&= \hat{p}^t\left(1-p\right) + p\left(1-\hat{p}^t\right) && \text{(As they are independent events)}
\end{aligned}$$

Exercise 4.1. Prove that

$$\mathbb{E}\left[\ell^{0\text{-}1}\left(\hat{y}^t, y^t\right)\right] \geq \min\left\{p, 1-p\right\}$$

Notice that the above suggests that the best option for us is to predict 1 as the outcome if $p > (1-p)$, and 0 otherwise. Indeed the result below shows that this is, in fact, true of not just a single time step t, but the entire online process altogether.

Theorem 4.1. For any algorithm $\mathcal{A}$, we have

$$\mathbb{E}\left[L^{0\text{-}1}\left(\mathcal{A}, \mathbf{y}\right)\right] \geq T \cdot \min\left\{p, 1-p\right\}.$$

Proof. For any algorithm $\mathcal{A}$, we have

$$\begin{aligned}
\mathbb{E}\left[L^{0\text{-}1}\left(\mathcal{A}, \mathbf{y}\right)\right] &= \mathbb{E}\left[\sum_{t=1}^{T} \ell^{0\text{-}1}\left(\mathcal{A}^t, y^t\right)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\ell^{0\text{-}1}\left(\mathcal{A}^t, y^t\right)\right] && \text{(By linearity of expectation)} \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\ell^{0\text{-}1}\left(\hat{y}^t, y^t\right)\right] \\
&\geq \sum_{t=1}^{T} \min\left\{p, 1-p\right\} && \text{(By Exercise 4.1).}
\end{aligned}$$

Note that the expectation above is over the randomness of the toss of the coin, as well as any internal randomness used by the algorithm, to make its prediction. □

We note that there exists a very simple algorithm that achieves this lower bound. Consider the algorithm which, if bias is greater than $1/2$, predicts only 1 (i.e. Heads) otherwise predicts 0 (i.e. Tails) (see Algorithm 2). It is easy to see that the expected cumulative mistake incurred by this algorithm achieves this lower bound. In the following, the notation $\mathcal{B}_p^T$ denotes the product distribution over T Bernoulli distributions, all with bias p.

Algorithm 2: OBP Benchmark $\mathcal{A}^*$

Let $y^* \leftarrow \mathbb{I}\left\{p > \frac{1}{2}\right\}$
for $t = 1, 2, \ldots, T$ **do**
    Predict y^*
end for

Lemma 4.2. $\underset{\mathbf{y}\sim\mathcal{B}_p^T}{\mathbb{E}}\left[L^{0\text{-}1}\left(\mathcal{A}^*, \mathbf{y}\right)\right] = T \cdot \min\left\{p, 1-p\right\}.$

Proof. Without loss of generality, let us assume that $p \leq 1-p$, i.e. $p \leq \frac{1}{2}$ (a similar analysis holds when $p > (1-p)$). Thus, $\mathcal{A}^*$ will always predict Tails for all t irrespective of stream history. So, $\mathcal{A}^*$ will make a mistake whenever the outcome is Heads. Hence, we have

$$\begin{aligned}
\underset{\mathbf{y}\sim\mathcal{B}_p^T}{\mathbb{E}}\left[L^{0\text{-}1}\left(\mathcal{A}^*, \mathbf{y}\right)\right] &= \sum_{t=1}^{T} \mathbb{E}\left[\ell^{0\text{-}1}\left(y^*, y^t\right)\right] \\
&= \sum_{t=1}^{T} \mathbb{P}\left[y^t \neq 0\right] \\
&= Tp \\
&= T \cdot \min\left\{p, 1-p\right\}
\end{aligned}$$

□

This motivates us to select $\mathcal{A}^*$ as the benchmark for our analysis. In the next section, we will look at an algorithm that can compete with this benchmark.

4 Following the Leader

Given the nature of the benchmark, which purely relies on whether $p > 1/2$ or not, a natural algorithm for the online coin toss prediction algorithm is one that tries to estimate p itself, in order to figure out which side of $1/2$ does it lie on.

This is interesting since it tells us how the online coin toss prediction problem is related to the online parameter estimation problem – solving the coin toss prediction problem, in some sense, requires us to estimate the bias of the coin (at least well enough to figure out whether it is greater or lesser than $1/2$).

Given the stochasticity of the coin tosses, it is tempting to use the counts of heads and tails seen so far to perform the prediction: if we have seen more heads till now, predict heads else tails. This is justified since if $p > 1/2$, we do expect to see more heads than tails. This intuitive policy gives us the `Majority` algorithm,

Algorithm 3: `Majority/FTL`

for $t = 1, 2, \ldots, T$ **do**
    Let $\hat{p}^t \leftarrow \frac{1}{t-1}\sum_{\tau=1}^{t-1} y^\tau$
    Predict $\hat{y}^t = \mathbb{I}\left\{\hat{p}^t > \frac{1}{2}\right\}$
    Receive y^t
end for

As discussed, the `Majority` algorithm counts the heads and tails seen so far and predicts, at every time step, whichever was more frequent till that time step.

4.1 Enter the Experts

There is another nice interpretation of the `Majority` algorithm in terms of *experts*. Imagine that we have experts giving us *advice*. The first expert $\mathfrak{E}^1$, always thinks the next toss is going to be Heads and advises us to predict $\hat{y}^t = 1$. The other $\mathfrak{E}^0$ which always predicts Tails (and tells us that we should predict $\hat{y}^t = 0$). At every time step, when we make a prediction, in some sense we have to believe one of these two experts.

Given this problem setting, we can reinterpret the `Majority` algorithm described above as a `Follow-the-leader` algorithm. At every time step, we simply believe the expert that has made minimum mistakes so far, or alternatively, gotten most of the predictions correct so far. The `Follow-the-leader` (or `FTL` in short) strategy is a very popular one in online learning de Rooij et al. (2014) and one that we will revisit later in the course.

Also note that if $p > 1/2$ then the benchmark is actually the "Head" expert i.e. $\mathcal{A}^* \equiv \mathfrak{E}^1$ and otherwise if $p \leq 1/2$ then $\mathcal{A}^* \equiv \mathfrak{E}^0$. Thus, the benchmark is always one of the experts and hence, online bit prediction problem also becomes an attempt to find the best expert. In fact, this is true of *prediction with expert advice* in general, which we will study in the next lecture.

4.2 A Mistake Bound for `FTL`

Coming back to the problem at hand, to obtain a *mistake bound* for the `FTL` strategy, we need to note the situations in which it makes a mistake. Let $\hat{p}^t := \frac{1}{t-1}\sum_{\tau=1}^{t-1} y^\tau$ as in Algorithm 3. Then there are two situations when `FTL` makes a mistake:

1. `FTL` disagrees with $\mathcal{A}^*$ at time t and makes a mistake i.e. $\mathcal{A}^*$ is correct at time t.
2. `FTL` agrees with $\mathcal{A}^*$ at time t but makes a mistake since $\mathcal{A}^*$ makes a mistake at time t.

We need not worry about mistakes of the second kind. This is because we are allowed to make as many mistakes as the benchmark does. Our aim is to show that we do not make too many more mistakes than the benchmark. Thus, our focus here would be to bound mistakes of the first kind. This is formalized below in a proof. We will use the standard notation to denote

$$L^{0\text{-}1}(\texttt{FTL}, \mathbf{y}) := \sum_{t=1}^{T} \ell^{0\text{-}1}(\texttt{FTL}^t, y^t).$$

Let $y^* := \mathbb{I}\left\{p > \frac{1}{2}\right\}$ denote the (constant) response of the benchmark $\mathcal{A}^*$. Also, let B^t be the "bad" event that `FTL` disagrees with $\mathcal{A}^*$ at time t i.e. $B^t := \mathbb{I}\left\{\hat{p}^t > \frac{1}{2}\right\} \neq y^*$. This gives us

$$\begin{aligned}
L^{0\text{-}1}(\texttt{FTL}, \mathbf{y}) - L^{0\text{-}1}(\mathcal{A}^*, \mathbf{y}) &= \sum_{t=1}^{T} \ell^{0\text{-}1}(\texttt{FTL}^t, y^t) - \ell^{0\text{-}1}(y^*, y^t) \\
&= \sum_{t=1}^{T} \left(\ell^{0\text{-}1}(\texttt{FTL}^t, y^t) - \ell^{0\text{-}1}(y^*, y^t)\right) \cdot \left(\mathbb{I}\left\{B^t\right\} + \mathbb{I}\left\{\neg B^t\right\}\right) \\
&= \sum_{t=1}^{T} \left(\ell^{0\text{-}1}(\texttt{FTL}^t, y^t) - \ell^{0\text{-}1}(y^*, y^t)\right) \cdot \mathbb{I}\left\{B^t\right\} \\
&\quad + \sum_{t=1}^{T} \left(\ell^{0\text{-}1}(y^*, y^t) - \ell^{0\text{-}1}(y^*, y^t)\right) \cdot \mathbb{I}\left\{\neg B^t\right\} \\
&= \sum_{t=1}^{T} \left(\ell^{0\text{-}1}(\texttt{FTL}^t, y^t) - \ell^{0\text{-}1}(y^*, y^t)\right) \cdot \mathbb{I}\left\{B^t\right\}
\end{aligned}$$

$$\leq \sum_{t=1}^{T} \mathbb{I}\left\{B^t\right\},$$

where in the third step, we have used the fact that $\neg B^t$ implies that $\texttt{FTL}^t = y^*$ and in the final step we have used the fact that $\ell^{0\text{-}1}(\texttt{FTL}^t, y^t) - \ell^{0\text{-}1}(y^*, y^t) \leq 1$ since the 0-1 loss function only takes the values $0, 1$.

Thus, an application of linearity of expectation tells us that

$$\begin{aligned}\mathbb{E}\left[L^{0\text{-}1}(\texttt{FTL}, \mathbf{y})\right] &\leq \mathbb{E}\left[L^{0\text{-}1}(\mathcal{A}^*, \mathbf{y})\right] + \sum_{t=1}^{T} \mathbb{E}\left[\mathbb{I}\left\{B^t\right\}\right] \\ &= \mathbb{E}\left[L^{0\text{-}1}(\mathcal{A}^*, \mathbf{y})\right] + \sum_{t=1}^{T} \mathbb{P}\left[B^t\right]\end{aligned}$$

Thus, all we need to do is bound $\mathbb{P}\left[B^t\right]$. Now, as the coin tosses $y^t, t = 1, 2, \ldots$ are i.i.d. realizations of the same Bernoulli random variable $\mathcal{B}(p)$, the strong law of large numbers tells us that

$$\mathbb{P}\left[\lim_{t\to\infty} \hat{p}^t = p\right] = 1$$

Thus, as $t \to \infty$, B^t does not occur almost surely i.e. $\lim_{t\to\infty} \mathbb{P}\left[B^t\right] = 0$. However, for finite t, we need to use finite sample versions of the strong law of large numbers. Given that we are working with Bernoulli random variables, the natural choice is the Chernoff bound. For the subsequent analysis, we will assume, without loss of generality, that $p \leq 1/2$. A similar result will hold for $p > 1/2$ as well.

Theorem 4.3 (Chernoff Bound - right tail)**.** (Wikipedia) Let X be a Bernoulli random variable with bias μ i.e. $\mathbb{E}X = \mu$. Let $X_1, X_2, \ldots, X_n$ be i.i.d. realizations of X. Let $\bar{X}_n = \frac{1}{n}\sum_i X_i$ denote the empirical expectation. Then, for every $\epsilon > 0$,

$$\mathbb{P}\left[\bar{X}_n \geq \mu + \epsilon\right] \leq \exp\left(-\mathrm{KL}(\mu + \epsilon \| \mu) \cdot n\right)$$

In order to use the above result, we need to quantify the separation between p and $\hat{p}^t$. The following lemma does this.

Lemma 4.4. $B^t \implies \hat{p}^t - p \geq \frac{1}{2} - p$

Proof. Recall that as assumed, we have $p \leq 1/2$. Then B^t implies that $\hat{p}^t$ must lie on the "other" side of $1/2$ i.e. $\hat{p}^t > 1/2$. This implies (see Figure 4.2) that $\hat{p}^t - p > 1/2 - p$ which establishes the claim. □

Corollary 4.5. $\mathbb{P}\left[B^t\right] \leq (4p(1-p))^{t/2}$

Proof. First of all notice that $\mathbb{E}\left[\hat{p}^t\right] = p$ by linearity of expectation and the fact that the coin tosses are all distributed as $\mathcal{B}(p)$. Next we notice that

$$\begin{aligned}\mathbb{P}\left[B^t\right] &\leq \mathbb{P}\left[\hat{p}^t - p \geq \frac{1}{2} - p\right] && \text{(Using Lemma 4.4)} \\ &= \mathbb{P}\left[\hat{p}^t \geq \mathbb{E}\left[\hat{p}^t\right] + \frac{1}{2} - p\right] \\ &\leq \exp\left(-\mathrm{KL}\left(\frac{1}{2} \,\middle\|\, p\right) \cdot t\right) && \text{(Using Theorem 4.3)} \\ &= (4p(1-p))^{t/2}.\end{aligned}$$

We note that the same result also holds if $p > 1/2$. Note that $4p(1-p) < 1$ for $p \neq 1/2$. □

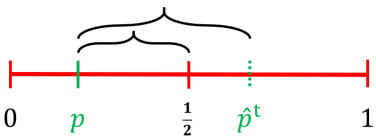

Figure 1: If $p \in \left[0, \frac{1}{2}\right]$ and $\hat{p}^t \in \left[\frac{1}{2}, 1\right]$, then the distance between p and $\hat{p}^t$ is at least $\frac{1}{2} - p$.

Theorem 4.6 (FTL Mistake Bound). $\mathbb{E}\left[L^{0\text{-}1}(\texttt{FTL}, \mathbf{y})\right] \leq \mathbb{E}\left[L^{0\text{-}1}(\mathcal{A}^*, \mathbf{y})\right] + \mathcal{O}(1)$

Proof. The previous analysis tells us that

$$\mathbb{E}\left[L^{0\text{-}1}(\texttt{FTL}, \mathbf{y})\right] \leq \mathbb{E}\left[L^{0\text{-}1}(\mathcal{A}^*, \mathbf{y})\right] + \sum_{t=1}^{T} \mathbb{P}\left[B^t\right]$$

Applying Corollary 4.5 proves the result upon application of the following inequality:

$$\sum_{t=1}^{T} (4p(1-p))^{t/2} \leq \int_0^T (4p(1-p))^{t/2}\, dt \leq \int_0^\infty (4p(1-p))^{t/2}\, dt = \frac{-2}{\ln(4p(1-p))}.$$

Note that $4p(1-p) \leq 1$ for all $p \in [0,1]$ so the right hand side is always a positive quantity. □

This shows us that the `Majority` or the `FTL` algorithm enjoys a constant expected regret against the optimal benchmark predictor $\mathcal{A}^*$ in terms of mistake bounds – it makes only constantly many more mistakes on expectation than the benchmark.

Remark 4.2 (High Confidence Bounds). The reader might be wondering why are we satisfied with obtaining an expected mistake bound when the above online process can very well have a large variance. Proofs of results on expectation are usually easier as well as cleaner. However, we hasten to assure the reader that the results obtained above (as well as those we would be obtaining in the future lectures) on expected mistake/regret bounds can, in most cases, be converted to high probability bounds, at the cost of $o(T)$ additional mistakes/regret. These techniques will be studied by us in the second part of the course.

5 Bracing for More Powerful Adversaries

The adversary that we dealt with in the bit prediction problem is a rather dull one – it passively tosses a coin and simply lets us know the result. Such adversaries are called *stochastic adversaries* and they play a crucial role in the design and analysis of stochastic learning and optimization algorithms. We will revisit them later in the second part of the course.

Stochastic adversaries are an example of *non-reactive* environments where the environment, in this case the adversary, is not affected by our "action" i.e. our prediction. However, we can, and will, deal with more "reactive" adversaries who participate in the online process more intimately.

The most general and uninhibited of these are *fully adaptive* adversaries which critically analyze each step taken by the learner before deciding the feedback. Such adversaries are typically also fully aware of the algorithm that the learner is using to make predictions and are frequently endowed with unbounded computational power.

Competing with such adversaries is, in general, very hard. For example, in the bit prediction problem, a fully adaptive adversary can always make us incur T mistakes by simply giving $y^t = 1 - \hat{y}^t$ as the feedback. Fully adaptive adversaries are thus, not interesting in the bit

prediction problem. They are, however, studied very deeply in online linear classification and regression tasks. We will ourselves revisit them soon.

A third class of adversaries, which are called *oblivious adversaries*, lie midway between non-reactive and fully adaptive adversaries. These adversaries are fully aware of the learner's algorithm and usually have unbounded computational power, but they decide their feedback before the online process begins.

An example is that of predicting a sequence of T coin tosses or bits where the bits $\mathbf{y} \in \{0,1\}^T$ have been decided in advance. An interesting question in this case is whether we can compete with the experts $\mathfrak{E}^0$ and $\mathfrak{E}^1$ in this new setting. In other words, can we get no more mistakes than the following?

$$\min\left\{\sum y_t, \sum (1-y_t)\right\}$$

At first glance we notice that oblivious adversaries are not reactive and do not care about how the online process evolves. However, the reader should notice (and indeed try to prove) that it is impossible to defeat such adversaries using deterministic algorithms. We will look at defeating such oblivious adversaries in the next lecture.

References

Steven de Rooij, Tim van Erven, Peter D. Grünwald, and Wouter M. Koolen. Follow the Leader If You Can, Hedge If You Must. *Journal of Machine Learning Research*, 15(1):1281–1316, 2014.

Wikipedia. Chernoff bound. `https://en.wikipedia.org/wiki/Chernoff_bound#Precise_statements_and_proofs`.