CSCI 659 Lecture 1

Fall 2022

Instructor: Haipeng Luo

1 Overview of Online Learning

Online learning (also known as online optimization, sequential decision making, etc.) has been playing a crucial role in machine learning and many real-life applications. Its foundation is supported by elegant and deep theory, which is the main topic of this course, but before even telling you what online learning is, let me start with a couple of impactful applications of online learning to convince you of its importance:

- ADAGRAD [Duchi et al., 2011], ADAM [Kingma and Ba, 2015], and several other most popular optimization algorithms for training machine learning models including neural nets, are derived from an online learning framework;
- regret minimization, the central goal of online learning, is arguably the most efficient and scalable way for solving games (e.g., finding Nash equilibria) — for example, it is the cornerstone for superhuman AI for Poker [Bowling et al., 2015], a game with 10^{13} states;
- multi-armed bandits, an online learning framework fundamental for studying exploration versus exploitation tradeoff, has been used in building recommendation systems in many tech companies [Agarwal et al., 2016];
- machine learning algorithms that preserve privacy are gaining increasing attention in recent years — it turns out that, theoretically speaking, private learning and online learning are equivalent [Alon et al., 2022].

Hopefully now you are convinced that online learning is important. Below let's start looking at several more concrete instances of online learning.

- spam detection (online classification/regression): At each time $t = 1, 2, \ldots$
  - receive an email $x_t \in \mathbb{R}^d$;
  - predict whether it is a spam $\hat{y}_t \in \{-1, +1\}$;
  - see its true label $y_t \in \{-1, +1\}$ (spam or not).
- sequential investment (universal portfolio): Start with capital W_1. At each day $t = 1, 2, \ldots$
  - decide $p_t \in \Delta(N) \stackrel{\text{def}}{=} \{p \in \mathbb{R}^N_+ : \sum_{i=1}^N p(i) = 1\}$ and invest $W_t p_t(i)$ on asset i;
  - at the end of the day observe relative prices $r_t \in \mathbb{R}^N_+$ such that the capital on asset i becomes $W_t p_t(i) r_t(i)$;
  - arrive at total capital $W_{t+1} = W_t \langle p_t, r_t \rangle$.
- aggregating weather prediction (the expert problem): At each day $t = 1, 2, \ldots$
  - obtain temperature predictions from N experts/models for today;
  - make the final prediction by randomly following an expert according to $p_t \in \Delta(N)$;
  - at the end of the day observe the loss of each model $\ell_t \in [0, 1]^N$.
- product recommendation (multi-armed bandits): At each time $t = 1, 2, \ldots$

  - randomly recommend one of the K products a to a customer visiting the website;
  - observe the loss of this product $\ell_t(a)$ (e.g. 0 if clicked, 1 otherwise), but not the losses for the other products.
- multiple-product recommendation (combinatorial bandits): At each time $t = 1, 2, \ldots$
  - randomly recommend k of the K products to a customer visiting the website;
  - observe the losses of the k recommended products but not the other ones.
- personalized product recommendation (contextual bandits): Given N policies $\pi^1, \ldots, \pi^N$, each of which is a mapping from $\mathcal{X}$ to $[K] \stackrel{\text{def}}{=} \{1, \ldots, K\}$. At each time $t = 1, 2, \ldots$
  - observe the contextual information $x_t \in \mathcal{X}$ of a customer (e.g. gender, IP address, purchase history, etc);
  - randomly select one of the N policies π_t and recommend product $\pi_t(x_t)$;
  - observe the loss of this product $\ell_t(\pi_t(x_t))$ but not the other ones.

2 A Unified Model

All of the examples in the last section can be (essentially) captured by a learning model called *Online Convex Optimization (OCO)*, first proposed by Zinkevich [2003]. OCO can be viewed as a game between a learner/player and an environment/adversary. Before the game starts, a fixed compact convex set Ω is given to the learner as the action space. The game then proceeds for T rounds for some integer T. At each round $t = 1, \ldots, T$,

1. the learner first picks a point $w_t \in \Omega$;
2. the environment then picks a convex loss function $f_t : \Omega \to \mathbb{R}$;
3. the learner suffers loss $f_t(w_t)$, and observes some information about f_t.

The table below summarizes how OCO captures all the aforementioned examples.

Problems	Ω	f_t	Observations
linear classification linear regression	e.g. $\{w : \|w\|_2 \le 1\}$	$f_t(w) = \ell(\langle w, x_t \rangle, y_t)$, e.g. logistic loss: $\ell(\hat{y}, y) = \ln(1 + e^{-\hat{y}y})$ or square loss: $\ell(\hat{y}, y) = (\hat{y} - y)^2$	x_y and y_t thus entire f_t
universal portfolio	$\Delta(N)$	$f_t(p) = -\ln(\langle p, r_t \rangle)$ (note the unboundedness)	r_t, thus entire f_t
the expert problem	$\Delta(N)$	$f_t(p) = \langle p, \ell_t \rangle$	ℓ_t, thus entire f_t
multi-armed bandits	$\Delta(K)$	$f_t(p) = \langle p, \ell_t \rangle$	only one entry of ℓ_t
combinatorial bandits	e.g. $\{w \in [0,1]^K : \sum_{i=1}^K w(i) = k\}$	$f_t(w) = \langle w, \ell_t \rangle$	k entries of ℓ_t
contextual bandits	$\Delta(N)$	$f_t(w) = \sum_{j=1}^N w(j)\ell_t(\pi^j(x_t))$ (note the loss structure among policies)	only one entry of ℓ_t

A couple of remarks are in order.

Why convexity? This is mainly because a) many problems are indeed convex and b) convexity allows efficient and provable algorithms. Of course, neural nets, the currently dominating machine learning models in many areas, are non-convex. Extending the theory of online learning to the non-convex regime is an important research direction, but note that even in the traditional "offline" setting, little is known currently.

Also note that for some problems, the action set is naturally discrete (e.g. all the bandit examples), but it can still be easily captured by OCO via convexifying the discrete action set (i.e., taking the convex hull).

Feedback model. As illustrated by these examples, there are several possible feedback models in OCO, roughly categorized into the following three:

- full-information setting: learner observes f_t (or sometimes just (sub)gradient $\nabla f_t(w_t)$);
- bandit setting: learner observes only $f_t(w_t)$;
- other partial information settings (sometimes not even the loss of the selected action).

The first half of this course will focus on the full full-information setting, while the second half focuses on the bandit setting.

Types of Environment. So far, we have not discussed how the environment decides the loss functions. Depending on its power, there are several possible settings (all of which allow the environment to have knowledge about the learner's algorithm, but not her randomness, ahead of time):

- stochastic setting: $f_1, \dots, f_T$ are i.i.d. samples of a fixed and unknown distribution;
- oblivious adversary: $f_1, \dots, f_T$ are arbitrary, but decided before the game starts (i.e. independent of the learner's actions);
- non-oblivious/adaptive adversary: for each t, f_t depends on $w_1, \dots, w_t$.

In this course we will mostly focus on the more general adversarial settings.

2.1 Goal of the Learner: Regret Minimization

The classical goal of OCO is to minimize the learner's (static) regret against the best fixed action in hindsight:

$$\mathcal{R}_T \stackrel{\text{def}}{=} \sum_{t=1}^{T} f_t(w_t) - \min_{w \in \Omega} \sum_{t=1}^{T} f_t(w).$$

Note that this could be a random variable where the randomness might come from the learner and/or the environment. Intuitively, this is measuring how much we "regret" for not always picking the best fixed action in retrospect. If the regret is sublinear in T, that is, $\mathcal{R}_T = o(T)$ (in which case the learner is often called "no-regret"), then $\lim_{T\to\infty} \mathcal{R}_T/T = 0$ and thus on average the learner is doing almost as well as the best fixed action in hindsight, which is a pretty strong guarantee. However, two very natural questions arise upon closer examination of this performance measure:

1. Why is it fair to compete with a fixed action while the learner can vary her actions over time? And why is it reasonable anyway given that the loss functions are changing over time and probably no single action is reasonably good overall?

For the first part of the question, note that competing with a fixed action is already highly nontrivial, since this fixed action is chosen with perfect knowledge of the entire loss functions, while the learner receives information (sometimes only partial information) about them on the fly. That said, also note that the regret can indeed be negative exactly because of this reason if the environment is benign.

The second part of the question is a valid concern, and there are many stronger regret measures proposed to exactly address this issue, including interval/switching/dynamic regret and internal/swap regret, some of which will be covered later in this course.

2. Why do we assume that the environment still chooses the same sequence of loss functions in retrospect while the learner has changed her behavior to always picking a fixed action?

First note that at least for a stochastic environment or an oblivious adversary, this assumption indeed holds (since the loss functions are chosen independent of the learner's decisions). However, this is indeed a valid concern for adaptive adversary, and a stronger regret measure called policy regret was proposed to exactly address this issue.

However, despite all these issues, **studying static regret is still extremely meaningful** for two reasons. First, static regret is the foundation of all other stronger regret measures. Algorithms for these stronger measures are most often designed by extending techniques from the static regret literature or

sometimes even by reducing the problem to minimizing static regret. Second, later in this course we will see examples in solving games via online learning, where the environment is definitely changing in an adaptive way, yet minimizing static regret still has strong implications for convergence to equilibra and optimal social welfare. Therefore, designing (static) regret minimization algorithm and proving their regret bounds has always been the central theme of online learning.

3 Connection to Statistical Learning

Before diving further into online learning, let's take a step back and discuss the connections and differences between online learning and the classical machine learning paradigm: *statistical learning*.

In statistical learning, a set of training examples $z_1, \dots, z_T \in \mathcal{Z}$ is given to the learner where each example z_t is an i.i.d. sample of some unknown distribution $\mathcal{D}$. Based on these training examples, the learner outputs a predictor $\bar{w} \in \Omega$ for some compact convex set Ω. For some loss function $\ell : \Omega \times \mathcal{Z} \to [0, 1]$, the *training error* of the learner is defined as $\frac{1}{T}\sum_{t=1}^{T} \ell(\bar{w}, z_t)$ while the *generalization error* is defined as $L(\bar{w}) \stackrel{\text{def}}{=} \mathbb{E}_{z\sim\mathcal{D}}\ell(\bar{w}, z)$. Note that this quantity is itself a random variable since $\bar{w}$ is random due to the randomness of the training set. The goal of the learner is usually to have small generalization error with high probability.

As one can see, distributional assumptions are fundamental in the definition of statistical learning. On the other hand, online learning does not necessarily assume any distributional distributions, which makes it a much more robust model. In fact, even if the data are entirely adversarial, which is indeed the case for applications such as spam detection or playing games, meaningful and strong guarantees can still be derived.

Another key advantage is that online learning algorithms are usually more memory-efficient, in the sense that they usually do not need to store data from the past. That is, at each round, a new data point is used to update the current state of the algorithm and then discarded. On the other hand, most statistical learning algorithms require storing the entire training set and touching each data point multiple times.

Finally and perhaps most importantly, one can in fact use an online learning algorithm to solve a statistical learning problem, meaning that online learning is only more general. This can be done via the following online-to-batch reduction [Cesa-Bianchi et al., 2004], which essentially lets an online learning algorithm make exactly one pass to the training set.

Algorithm 1: Online-to-Batch Reduction

Input: training set $\{z_1, \dots, z_T\}$, an online learning algorithm $\mathcal{A}$ with action space Ω
for $t = 1, \dots, T$ **do**
  let w_t be the output of $\mathcal{A}$ for this round
  feed $\mathcal{A}$ with loss function $f_t(w) = \ell(w, z_t)$
Output (Option I): uniformly at random sample $\bar{t}$ from $\{1, \dots, T\}$, then $\bar{w} = w_{\bar{t}}$
Output (Option II): $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$ if $L(w)$ is convex in w

The reduction enjoys the following guarantee.

Theorem 1. *Suppose that $\mathcal{A}$ enjoys the following regret bound $\mathbb{E}[\mathcal{R}_T] \le B(T)$ for some function B. Then the generalization error of the output $\bar{w}$ of Algorithm 1 satisfies*

$$\mathbb{E}[L(\bar{w})] \le L(w^\star) + \frac{B(T)}{T}$$

where $w^\star \in \operatorname{argmin}_{w\in\Omega} L(w)$ is the optimal predictor.

Proof of Theorem 1. First focus on Option I. By definitions, we have (pay close attention to what randomness each expectation $\mathbb{E}[\cdot]$ is with respect to)

$$\mathbb{E}[L(\bar{w})] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[L(w_t)] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\mathbb{E}_{z\sim\mathcal{D}}\ell(w_t, z)]$$

$$= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\ell(w_t, z_t)] \le \mathbb{E}\left[\min_{w\in\Omega}\frac{1}{T}\sum_{t=1}^{T}\ell(w, z_t)\right] + \frac{B(T)}{T}$$
$$\le \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\ell(w^\star, z_t)] + \frac{B(T)}{T}$$
$$= \frac{1}{T}\sum_{t=1}^{T}L(w^\star) + \frac{B(T)}{T} = L(w^\star) + \frac{B(T)}{T}.$$

For Option II, simply notice that by the convexity of L and Jensen's inequality:

$$L(\bar{w}) = L\left(\frac{1}{T}\sum_{t=1}^{T}w_t\right) \le \frac{1}{T}\sum_{t=1}^{T}L(w_t).$$

The rest of the proof is identical to Option I. □

Therefore, as long as $\mathcal{A}$ is no-regret (i.e. $B(T) = o(T)$), then the generalization error of $\bar{w}$ is arbitrarily close to the optimal error when the sample size T is large enough. For example, for many problems we can show $B(T) = \mathcal{O}(\sqrt{T})$, which implies a convergence rate of $1/\sqrt{T}$ to the optimal generalization error and is often known to be optimal. We note that although we present an in-expectation guarantee here, the same holds up to an additional $\mathcal{O}(\sqrt{\ln(1/\delta)/T})$ term with probability at least $1-\delta$ by directly applying certain concentration inequalities (see e.g. [Luo, 2017]).

This illustrates that online learning algorithms can be used for traditional statistical learning problems as well, which is also the reason why algorithms such as AdaGrad and Adam are all derived from the OCO setting, but mostly used in training models with a fixed training set in practice.

Question. *Why making only one pass of the training set is important to Algorithm 1? Can you identify the step in the proof that will break as long as one training example is passed to $\mathcal{A}$ more than once?*

4 The Expert Problem and the Hedge Algorithm

Now we start discussing how to solve online learning problems. We defer the general OCO case to the next lecture, and focus on one special case, the expert problem, in the rest of this lecture. It turns out that this problem plays some fundamental role in online learning (and other related topics), as we will see soon in this course. The expert problem is originated from [Freund and Schapire, 1997]. Recall that in this problem, at each round t the learner decides a distribution $p_t \in \Delta(N)$ over N experts, and then suffers loss $\langle p_t, \ell_t\rangle$ for some loss vector $\ell_t \in [0,1]^N$ decided by the environment. How do we ensure sublinear regret in this problem?

The first naive algorithm that comes to one's mind is probably the *follow the leader* (FTL) approach, which puts all the weights to the current best expert $\operatorname{argmax}_{i\in[N]} -\sum_{s=1}^{t-1}\ell_s(i)$. However, it is not difficult to see that such an approach in fact could suffer linear regret in the worst case: simply consider the case where $N=2$, $\ell_1 = (0.5, 0)$, and for $t \ge 2$, ℓ_t alternates between $(0,1)$ and $(1,0)$.

It turns out that, however, simply replacing the "max" in this naive algorithm by some "softmax" would change the situation greatly. In fact, this leads to the the classical algorithm called Hedge [Littlestone and Warmuth, 1994, Freund and Schapire, 1997]. The same algorithm is also known as multiplicative weights update, exponential weights, and many others, and was discovered independently in many different areas. We present the pseudocode below.

Algorithm 2: Hedge

Input: learning rate $\eta > 0$ (also called step size, temperature, etc.)
Initialization: let $L_0 \in \mathbb{R}^N$ be the all-zero vector
for $t = 1, \dots, T$ **do**
  compute $p_t \in \Delta(N)$ such that $p_t(i) \propto \exp(-\eta L_{t-1}(i))$
  play p_t and observe loss vector $\ell_t \in [0,1]^N$
  update $L_t = L_{t-1} + \ell_t$

We will discuss more intuition of this algorithm in the next lecture when putting it into a more general framework, but it is clear that this algorithm is at least doing something sensible: the better an expert looks based on the past cumulative loss, the more weight we put on this expert, but at the same time we do not act as extremely as the FTL approach and always make sure that every expert gets some non-zero weight. We also point out that this is definitely not the only algorithm for this problem, and there are many other reasonable ways to spread the weights and ensure low regret.

The regret for this algorithm can be written as:

$$\mathcal{R}_T = \sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{p \in \Delta(N)} \sum_{t=1}^{T} \langle p, \ell_t \rangle = \sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \sum_{t=1}^{T} \ell_t(i^\star)$$

where $i^\star \in \operatorname{argmin}_i \sum_{t=1}^{T} \ell_t(i)$ is the best expert in hindsight. Below we will prove a bound for any loss sequence $\ell_1, \ldots, \ell_T$.

Theorem 2. *Hedge with learning rate η guarantees for any loss sequence $\ell_1, \ldots, \ell_T \in [0,1]^N$:*

$$\mathcal{R}_T \le \frac{\ln N}{\eta} + \eta \sum_{t=1}^{T} \sum_{i=1}^{N} p_t(i) \ell_t^2(i) \tag{1}$$

$$\le \frac{\ln N}{\eta} + T\eta, \tag{2}$$

which is of order $\mathcal{O}(\sqrt{T \ln N})$ if η is optimally set to $\sqrt{(\ln N)/T}$.

There are many different proofs that lead to bound (2), but we will present a "potential-based" proof that uses bound (1) as an intermediate step, which will turn out to be very useful later on.

Proof. Define potential $\Phi_t = \frac{1}{\eta} \ln \left(\sum_{i=1}^{N} \exp(-\eta L_t(i)) \right)$. First consider how much the potential can increase between two consecutive rounds:

$$\begin{aligned}
\Phi_t - \Phi_{t-1} &= \frac{1}{\eta} \ln \left(\frac{\sum_{i=1}^{N} \exp(-\eta L_t(i))}{\sum_{i=1}^{N} \exp(-\eta L_{t-1}(i))} \right) = \frac{1}{\eta} \ln \left(\frac{\sum_{i=1}^{N} \exp(-\eta L_{t-1}(i)) \exp(-\eta \ell_t(i))}{\sum_{i=1}^{N} \exp(-\eta L_{t-1}(i))} \right) \\
&= \frac{1}{\eta} \ln \left(\sum_{i=1}^{N} p_t(i) \exp(-\eta \ell_t(i)) \right) \\
&\le \frac{1}{\eta} \ln \left(\sum_{i=1}^{N} p_t(i) \left(1 - \eta \ell_t(i) + \eta^2 \ell_t^2(i)\right) \right) && (e^{-y} \le 1 - y + y^2 \text{ for all } y \ge -1) \\
&= \frac{1}{\eta} \ln \left(1 - \eta \langle p_t, \ell_t \rangle + \eta^2 \sum_{i=1}^{N} p_t(i) \ell_t^2(i) \right) \\
&\le -\langle p_t, \ell_t \rangle + \eta \sum_{i=1}^{N} p_t(i) \ell_t^2(i). && (\ln(1+y) \le y)
\end{aligned}$$

Summing over t, telescoping, and rearranging gives

$$\begin{aligned}
\sum_{t=1}^{T} \langle p_t, \ell_t \rangle &\le \Phi_0 - \Phi_T + \eta \sum_{t=1}^{T} \sum_{i=1}^{N} p_t(i) \ell_t^2(i) \\
&\le \frac{\ln N}{\eta} - \frac{1}{\eta} \ln \left(\exp(-\eta L_T(i^\star)) \right) + \eta \sum_{t=1}^{T} \sum_{i=1}^{N} p_t(i) \ell_t^2(i) \\
&\le \frac{\ln N}{\eta} + L_T(i^\star) + \eta \sum_{t=1}^{T} \sum_{i=1}^{N} p_t(i) \ell_t^2(i),
\end{aligned}$$

which proves Eq. (1). Eq. (2) follows immediately by the boundedness of losses. □

One might wonder how to come up with such a proof and the specific form of "potential" that seems to come from nowhere. We will not go deep into this question, but only point out that a) the potential does come from somewhere and in fact can be "derived" in a sense from a principled framework, and b) in the next lecture we will discuss a different and more general and intuitive proof.

Question. *Does Theorem 2 hold for an oblivious adversary or an adaptive adversary? If the learner randomly selects an expert i_t according to p_t at each time t, and we measure the regret by $\sum_{t=1}^T \ell_t(i_t) - \sum_{t=1}^T \ell_t(i^\star)$, can we still obtain a similar regret bound (in expectation or with high probability)? If so, does it hold for an oblivious adversary or an adaptive adversary?*

5 Lower bound for the Expert Problem

One important thing to note here is that the regret of Hedge has only logarithmic dependence on N, which is extremely important for problems with a huge number of experts as we will discuss in future lectures. But even if this regret bound already looks very good, how do we know whether we can do even better or not? In general, how can we tell whether a regret upper bound is satisfactory or not? The notion of minimax regret can be used to answer this question exactly. Intuitively, minimax regret is the smallest possible worst-case regret one can achieve. For example, the minimax regret of the expert problem with an oblivious adversary can be defined as

$$\min_{\mathcal{A}} \max_{\ell_1,\dots,\ell_T} \mathbb{E}_{\mathcal{A}}[\mathcal{R}_T]$$

where $\mathcal{A}$ is any legitimate expert algorithm and $\mathbb{E}_{\mathcal{A}}$ highlights the randomness with respect to the algorithm itself. Note that $\mathcal{R}_T$ depends on both $\mathcal{A}$ and all the losses even if the dependence is not explicitly spelled out. The existence of the Hedge algorithm already shows that

$$\min_{\mathcal{A}} \max_{\ell_1,\dots,\ell_T} \mathbb{E}_{\mathcal{A}}[\mathcal{R}_T] \le 2\sqrt{T \ln N}.$$

If we can further show a $\Omega(\sqrt{T \ln N})$ lower bound, that is, no algorithm can do better than $\mathcal{O}(\sqrt{T \ln N})$ in the worst case, then we call this bound the minimax optimal regret. The following theorem essentially shows that this is indeed the case (and thus Hedge is minimax optimal).

Theorem 3. *For any algorithm, we have*

$$\sup_{T,N} \max_{\ell_1,\dots,\ell_T} \frac{\mathbb{E}_{\mathcal{A}}[\mathcal{R}_T]}{\sqrt{T \ln N}} \ge \frac{1}{\sqrt{2}}.$$

How do we prove such a lower bound? It seemingly requires us to construct a worst-case loss sequence for any given algorithm, which is quite difficult given that an algorithm is a very complex object. Instead, a common technique is to construct some randomized environment, and then argue that all algorithms have to suffer certain regret in expectation, which in turn proves that for every algorithm, there exists a bad sequence leading to such regret, without even explicitly constructing it.

Proof. Let $\mathcal{D}$ be the uniform distribution over $\{0, 1\}$. We have

$$\begin{aligned}
\max_{\ell_1,\dots,\ell_T} \mathbb{E}_{\mathcal{A}}[\mathcal{R}_T] &\ge \mathbb{E}_{\ell_1,\dots,\ell_T \overset{iid}{\sim} \mathcal{D}^N} \mathbb{E}_{\mathcal{A}}[\mathcal{R}_T] \\
&= \sum_{t=1}^T \mathbb{E}_{\mathcal{A},\ell_1,\dots,\ell_t}[\langle p_t, \ell_t \rangle] - \mathbb{E}_{\ell_1,\dots,\ell_T}\left[\min_{i\in[N]} \sum_{t=1}^T \ell_t(i)\right] \\
&= \sum_{t=1}^T \mathbb{E}_{\mathcal{A},\ell_1,\dots,\ell_{t-1}} \langle p_t, \mathbb{E}_{\ell_t}[\ell_t] \rangle - \mathbb{E}_{\ell_1,\dots,\ell_T}\left[\min_{i\in[N]} \sum_{t=1}^T \ell_t(i)\right] \\
&= T/2 - \mathbb{E}_{\ell_1,\dots,\ell_T}\left[\min_{i\in[N]} \sum_{t=1}^T \ell_t(i)\right] \\
&= \mathbb{E}_{\ell_1,\dots,\ell_T}\left[\max_{i\in[N]} \sum_{t=1}^T (\tfrac{1}{2} - \ell_t(i))\right]
\end{aligned}$$

$$= \frac{1}{2}\mathbb{E}_{\sigma_1,\ldots,\sigma_T}\left[\max_{i\in[N]}\sum_{t=1}^{T}\sigma_t(i)\right],$$

where $\sigma_t(i)$ for $i \in [N], t \in [T]$ are i.i.d. Rademacher random variables (i.e. -1 with probability 0.5 and 1 with probability 0.5). Note that at this point we have converted the problem to calculating a simple quantity that is independent of the algorithm and also has no sequential structure. Using the following standard result from probability theory (see for example [Cesa-Bianchi and Lugosi, 2006, Lemmas A.11 and A.12]) completes the proof.

$$\lim_{T\to\infty}\lim_{N\to\infty}\frac{\mathbb{E}_{\sigma_1,\ldots,\sigma_T}\left[\max_{i\in[N]}\sum_{t=1}^{T}\sigma_t(i)\right]}{\sqrt{T\ln N}} = \sqrt{2}.$$

□

Note that this lower bound is asymptotic — it does not tell us for a specific N and T, what the optimal regret is. For a non-asymptotic lower bound, see for example [Orabona and Pál, 2015].

References

A. Agarwal, S. Bird, M. Cozowicz, L. Hoang, J. Langford, S. Lee, J. Li, D. Melamed, G. Oshri, O. Ribas, S. Sen, and A. Slivkins. A multiworld testing decision service. *arXiv:1606.03966*, 2016.

Noga Alon, Mark Bun, Roi Livni, Maryanthe Malliaris, and Shay Moran. Private and online learnability are equivalent. *ACM Journal of the ACM (JACM)*, 2022.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

Haipeng Luo. Lecture notes 1, introduction to online learning, 2017. URL `https://haipeng-luo.net/courses/CSCI699/lecture1.pdf`.

Francesco Orabona and Dávid Pál. Optimal non-asymptotic lower bound on the minimax regret of learning with expert advice. *arXiv preprint arXiv:1511.02176*, 2015.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.