Theoretical Machine Learning
Lecture 6

Instructor: Haipeng Luo

1 Online Regression

Recall that in previous lectures, for a supervised learning problem with $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$, we derived an upper bound on the value of online learning $\mathcal{V}^{\text{seq}}(\mathcal{F}, n)$ in terms of the sequential Rademacher complexity of the function class $\mathcal{R}^{\text{seq}}(f)$. We have also discussed the exact analogue of VC theory for binary classification $\mathcal{Y} = \{-1, +1\}$, with Littlestone dimension being the key complexity measure. Next, we briefly go through the same analogue for regression problems with $\mathcal{Y} = [-1, +1]$.

The first key concept is still covering. We say that a set V of $[-1, +1]$-valued trees is an *α-cover* (with respect to ℓ_p norm) of $\mathcal{F}$ for a $\mathcal{X}$-valued tree $\boldsymbol{x}$ if

$$\forall f \in \mathcal{F}, \forall \boldsymbol{\epsilon} \in \{-1, +1\}^n, \exists \boldsymbol{v} \in V, \text{s.t.} \left(\frac{1}{n}\sum_{t=1}^{n} |f(\boldsymbol{x}_t(\boldsymbol{\epsilon})) - \boldsymbol{v}_t(\boldsymbol{\epsilon})|^p\right)^{1/p} \leq \alpha.$$

Note that the order of the quantifiers $\forall \boldsymbol{\epsilon}$ and $\exists v$ is consistent with the online classification case discussed last time, and also that this serves as an exact analogue of α-cover for the statistical learning setting. The α-covering number $\mathcal{N}_p(F|_{\boldsymbol{x}}, \alpha)$ is simply the size of the minimum α-cover (with respect to ℓ_p norm). As always, by definition $\mathcal{N}_p(F|_{\boldsymbol{x}}, \alpha)$ is non-increasing in α, and the normalization is such that $\mathcal{N}_p(F|_{\boldsymbol{x}}, \alpha)$ is non-decreasing in p.

An α-cover naturally allows us to move from an infinite class to a finite class and derive the following bound on the sequential Rademacher complexity:

Theorem 1. *For any $\mathcal{X}$-valued tree $\boldsymbol{x}$ and $\mathcal{F} \subset [-1, +1]^{\mathcal{X}}$, we have*

$$\widehat{\mathcal{R}}^{\text{seq}}(\mathcal{F}; \boldsymbol{x}) \leq \inf_{\alpha \geq 0}\left(\alpha + \sqrt{\frac{2\ln \mathcal{N}_1(F|_{\boldsymbol{x}}, \alpha)}{n}}\right).$$

Proof. Fix any α. Let V be an α-cover of $F|_{\boldsymbol{x}}$ with size $\mathcal{N}_p(F|_{\boldsymbol{x}}, \alpha)$ and $\boldsymbol{v}^{f,\boldsymbol{\epsilon}} \in V$ be the tree that "certifies" the cover. We then have

$$\begin{aligned}
\widehat{\mathcal{R}}^{\text{seq}}(\mathcal{F}; \boldsymbol{x}) &= \frac{1}{n}\mathbb{E}\left[\sup_{f\in\mathcal{F}}\sum_{t=1}^{n}\epsilon_t f(\boldsymbol{x}_t(\boldsymbol{\epsilon}))\right] \\
&= \frac{1}{n}\mathbb{E}\left[\sup_{f\in\mathcal{F}}\sum_{t=1}^{n}\epsilon_t (f(\boldsymbol{x}_t(\boldsymbol{\epsilon})) - \boldsymbol{v}_t^{f,\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}) + \boldsymbol{v}_t^{f,\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}))\right] \\
&\leq \frac{1}{n}\mathbb{E}\left[\sup_{f\in\mathcal{F}}\sum_{t=1}^{n}\epsilon_t (f(\boldsymbol{x}_t(\boldsymbol{\epsilon})) - \boldsymbol{v}_t^{f,\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}))\right] + \frac{1}{n}\mathbb{E}\left[\sup_{f\in\mathcal{F}}\sum_{t=1}^{n}\epsilon_t \boldsymbol{v}_t^{f,\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})\right] \\
&\leq \alpha + \frac{1}{n}\mathbb{E}\left[\sup_{v\in V}\sum_{t=1}^{n}\epsilon_t \boldsymbol{v}(\boldsymbol{\epsilon})\right] \leq \alpha + \sqrt{\frac{2\ln \mathcal{N}_1(F|_{\boldsymbol{x}}, \alpha)}{n}},
\end{aligned}$$

where the last step uses the finite class bound. □

Using similar chaining techniques, one can also tighten the bound as (proof omitted):

Theorem 2. *For any $\mathcal{X}$-valued tree $\boldsymbol{x}$ and $\mathcal{F} \subset [-1,+1]^{\mathcal{X}}$, we have*

$$\widehat{\mathcal{R}}^{\text{seq}}(\mathcal{F};\boldsymbol{x}) \le \inf_{0\le\alpha\le 1}\left(4\alpha + \frac{12}{\sqrt{n}}\int_{\alpha}^{1}\sqrt{\ln \mathcal{N}_2(F|_{\boldsymbol{x}},\delta)}d\delta\right).$$

At this point, it is probably not surprising that there is again a combinatorial parameter that provides a nice upper bound on the covering number. To introduce such a parameter, we first generalize the shattering concept: a tree $\boldsymbol{x}$ is α-shattered by a class $\mathcal{F} \in [-1,+1]^{\mathcal{X}}$ if there exists a $[-1,+1]$-valued tree $\boldsymbol{y}$ (called the witness) such that

$$\forall \boldsymbol{\epsilon} \in \{-1,+1\}^n, \exists f \in \mathcal{F}, \text{s.t. } \epsilon_t(f(\boldsymbol{x}_t(\boldsymbol{\epsilon})) - \boldsymbol{y}_t(\boldsymbol{\epsilon})) \ge \alpha/2 \text{ holds for all } t = 1,\dots,n.$$

The (sequential) fat-shattering dimension of $\mathcal{F}$ at scale α, denoted by $\text{sfat}(\mathcal{F},\alpha)$, is then defined as the depth of the largest tree that can be α-shattered by $\mathcal{F}$. The following theorem shows the connection between covering number and fat-shattering dimension (proof omitted):

Theorem 3. *For any $\alpha > 0$, tree $\boldsymbol{x}$ of depth n, and function class $\mathcal{F} \subset [-1,+1]^{\mathcal{X}}$, we have $\mathcal{N}_\infty(F|_{\boldsymbol{x}},\alpha) \le \left(\frac{2en}{\alpha}\right)^{\text{sfat}(\mathcal{F},\alpha)}$.*

1.1 Example 1: non-decreasing functions

We have shown that in the statistical learning setting the class of all non-decreasing $[-1,+1]$-valued functions defined over $\mathbb{R}$ has α-covering number $(n+1)^{1/\alpha}$ and fat-shattering dimension $4/\alpha$. What about the online setting? We argue that no meaningful covering number or fat-shattering dimension can be derived, since the problem is simply not online learnable. To see this, first recall the tree we constructed last time to prove that finite Littlestone dimension is necessary for online learnability: an infinite $[0,1]$-valued tree $\boldsymbol{x}$ with root being $1/2$, and the left child and right child of a node with value a at level t being $a - \frac{1}{2^{t+1}}$ and $a + \frac{1}{2^{t+1}}$ respectively.

The environment is also almost the same: at time t, picks $(x_t, y_t) = (\boldsymbol{x}_t(-\boldsymbol{\epsilon}), \epsilon_t)$ where $\epsilon_1,\dots,\epsilon_n$ are i.i.d. Rademacher random variables (note the opposite sign of the path compared to last time). Let the loss of the problem be the square loss $\ell(f,(x,y)) = (f(x)-y)^2$. Then no matter how the algorithm behaves (properly or improperly), because y_t is $+1$ or -1 with equal probability, the square loss of the learner is always at least $\frac{1}{2}\min_{\widehat{y}}\left((\widehat{y}-1)^2 + (\widehat{y}+1)^2\right) = 1$. On the other hand, by the construction of the tree, we always have $x_t \le x_{t'}$ for any t and t' such that $y_t = -1$ and $y_{t'} = +1$, and therefore we can always find a non-decreasing function that passes all the points and has zero square loss. This makes the learner suffer linear regret.

We remark without going into details that, nevertheless, the problem becomes learnable in the so-called "transductive" setting, where the set of $\{x_1,\dots,x_T\}$ is known to the learner ahead of the time (but not the order of $x_{1:T}$ nor the outcomes $y_{1:T}$ of course).

1.2 Example 2: linear functions

Recall the linear class defined by: $\mathcal{X} = B_q^d$, $\mathcal{F} = \left\{f_\theta(x) = \langle\theta, x\rangle \mid \theta \in B_p^d\right\}$ for some $p \ge 1$ and $q \ge 1$ such that $1/p + 1/q = 1$. We have shown that this class admits a *pointwise* α-cover $\mathcal{H}$ of size $\left(\frac{2}{\alpha}+1\right)^d$ (Proposition 2 of Lecture 3). It is clear that the projection $\mathcal{H}|_{\boldsymbol{x}}$ of this pointwise cover for a tree $\boldsymbol{x}$ is an α-cover for $F|_{\boldsymbol{x}}$, and thus $\mathcal{N}_\infty(F|_{\boldsymbol{x}},\alpha) \le \left(\frac{2}{\alpha}+1\right)^d$. According to previous calculations, the Dudley entropy integral bound gives $\mathcal{R}^{\text{seq}}(\mathcal{F}) \le \mathcal{O}(\sqrt{d/n})$. So linear class is online learnable, at the same rate as in the statistical learning setting.

Discussions on bridging classification and regression. While linear functions are online learnable for regression problems (as long as the loss is Lipschitz), as discussed in the last lecture, the class of linear classifiers $\mathcal{F} = \left\{f_{\theta,b}(x) = \text{sign}\left(\langle x,\theta\rangle + b\right) \mid \theta \in \mathbb{R}^d, b \in \mathbb{R}\right\}$ is not online learnable with respect to the 0-1 loss. However, what is usually done in practice is to reparametrize the binary function class as a real-valued class and then replace 0-1 loss with some surrogate loss that enjoys some nicer properties and is online learnable. Specifically, most binary classifier class takes the form $\mathcal{H} = \{\text{sign}(f(\cdot)) \mid f \in \mathcal{F}\}$ for some real-valued class $\mathcal{F}$ predicting some "score", so that the

sign of the score predicts the label and the magnitude of the score represents the "confidence" of the prediction. The class of linear classifiers defined above is certainly of this form.

In this case, there is no difference is picking $\mathcal{F}$ or $\mathcal{H}$ as the decision space and reference class for our problem. If we pick $\mathcal{F}$ instead, then the 0-1 loss becomes $\ell(f,(x,y)) = \mathbf{1}\{yf(x) \le 0\}$ and the problem involves dealing with a real-valued function class. The problem is of course still not online learnable if $\text{Ldim}(\mathcal{H}) = \infty$ since all we have done is to represent the same problem slightly differently. However, we could now choose another loss function $\ell'(f,(x,y))$ that is an upper bound of the 0-1 loss and is online learnable as the surrogate for optimizing 0-1 loss. Clearly we then have

$$\sum_{t=1}^{T} \mathbf{1}\{y_t f_t(x_t) \le 0\} \le \sum_{t=1}^{T} \ell'(f_t,(x_t,y_t)) = \inf_{f\in\mathcal{F}} \sum_{t=1}^{T} \ell'(f,(x_t,y_t)) + \text{Reg}(\mathcal{F},n),$$

where $\text{Reg}(\mathcal{F},n)$ is defined in terms of the surrogate loss ℓ'. Since $\mathbb{E}[\text{Reg}(\mathcal{F},n)/n]$ goes to zero, we know that the average number of mistakes of the learner is arbitrarily close to the average surrogate loss of the best function from the reference class. This circumvents the (frustrating) impossibility result of learning 0-1 loss in the online setting beyond simple classes.

In fact, this is usually what we do in the statistical learning setting as well, even though we can learn any class with finite VC dimension with respect to the 0-1 loss, via ERM. The reason is computational — finding ERM with respect to 0-1 loss is usually NP-hard while many surrogate losses, especially those that are convex, admit highly efficient algorithms for finding an ERM.

Typical surrogate losses include the hinge loss $\ell(f,(x,y)) = \max\{1-yf(x),0\}$, the logistic loss $\ell(f,(x,y)) = \log(1+e^{-yf(x)})$, and many more. Note that both losses are 1-Lipschitz (with respect to $f(x)$), and thus if $\mathcal{F}$ is the linear class, the problem is indeed online learnable according to previous discussions, giving a meaningful bound on the error rate of the learner.

Summary. For regression problems with a G-Lipschitz loss, we have derived the following

$$\begin{aligned}
\mathcal{V}^{\text{seq}}(\mathcal{F},n) &\le \sup_{\mathcal{P}\in\Delta(\mathcal{Z}^n)} \mathop{\mathbb{E}}_{z_{1:n}\sim\mathcal{P}} \left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{t=1}^{n} \left(\mathbb{E}_{z_t'\sim\mathcal{P}(\cdot|z_{1:t-1})}[\ell(f,z_t')] - \ell(f,z_t)\right)\right] \\
&\le 2\mathcal{R}^{\text{seq}}(\ell(\mathcal{F})) \le \mathcal{O}\left(G\ln^{3/2} n\right)\cdot \mathcal{R}^{\text{seq}}(\mathcal{F}) \\
&\le \mathcal{O}\left(G\ln^{3/2} n\right)\cdot \sup_{\boldsymbol{x}} \inf_{0\le\alpha\le 1}\left(4\alpha + \frac{12}{\sqrt{n}}\int_{\alpha}^{1}\sqrt{\ln \mathcal{N}_2(F|_{\boldsymbol{x}},\delta)}d\delta\right) \\
&\le \mathcal{O}\left(G\ln^{3/2} n\right)\cdot \inf_{0\le\alpha\le 1}\left(4\alpha + \frac{12}{\sqrt{n}}\int_{\alpha}^{1}\sqrt{\text{sfat}(\mathcal{F},\delta)\ln\left(\frac{2en}{\delta}\right)}d\delta\right).
\end{aligned}$$

It can be argued that this sequence of upper bounds is also tight and the sequential fat-shattering dimension is in some sense the right complexity measure. At this point, we have finished the discussions of statistical and online learnability for both classification and regression, which concludes the first main topic of this course.

2 Algorithms for Finite Classes

Next, we will move on to the second main topic of this course: algorithm design for online learning. Recall that for statistical learning, all the problems we have discussed are learnable simply via ERM, if they are learnable at all. But for online learning, we have only characterized what determines learnability, and if the problem is learnable, we do not know how to learn yet. To think about how to design algorithms, we first recall the learning protocol:

For each $t = 1,\dots,n$,

1. learner (possibly randomly) selects $\widehat{y}_t \in \mathcal{D}$;
2. simultaneously the environment selects $z_t \in \mathcal{Z}$;
3. the learner suffers $\ell(\widehat{y}_t, z_t)$ and observes z_t.

Recall that $\mathcal{D}$ is the decision space of the learner, which is the same as the reference class $\mathcal{F}$ if the learner is proper, or a superset of $\mathcal{F}$ if the learner is improper (and thus more powerful). For an oblivious environment, z_t does not depend on the learner's decisions and can be equivalently seen as chosen ahead of the time before the learning protocol starts. On the other hand, for an adaptive environment, z_t could depend on $\widehat{y}_{1:t-1}$.

Since the first learnable class we showed is finite class, we start with the case when $\mathcal{F}$ is finite. Without loss of generality assume the value of the loss is always in $[0, 1]$, then based on previous discussions there exists an algorithm with

$$\mathbb{E}[\mathrm{Reg}(\mathcal{F}, n)] = \mathbb{E}\left[\sum_{t=1}^{n} \ell(\widehat{y}_t, z_t) - \inf_{f\in\mathcal{F}} \sum_{t=1}^{n} \ell(f, z_t)\right] \le \mathcal{O}\left(\sqrt{n \ln |\mathcal{F}|}\right). \tag{1}$$

Finding an algorithm with this guarantee will be our goal.

2.1 Warm-up

As the first step, we consider the special case of binary classification with 0-1 loss (so $\mathcal{Z} = \mathcal{X} \times \{-1, +1\}$ and $\mathcal{F} \subset \{-1, +1\}^{\mathcal{X}}$), and make a strong *realizable* assumption which posits that there is always a perfect classifier in $\mathcal{F}$, that is, $\inf_{f\in\mathcal{F}} \sum_{t=1}^{n} \mathbf{1}\{f(x_t) \neq y_t\} = 0$. This can be seen as an assumption on the expressiveness of $\mathcal{F}$ or equivalently as a constraint on the environment. What is a reasonable algorithm in this case?

An obvious solution is to pick $\widehat{y}_t$ to be any classifier in $\mathcal{F}$ that has made no mistakes yet so far. Then, whenever the learner makes a mistake, she has one less available choices. Since there is always a perfect classifier in $\mathcal{F}$, the learner makes at most $|\mathcal{F}| - 1$ mistakes, which is also her regret. This regret bound is better than Equation (1) in terms of the dependence in n (in fact, it is independent of n), but exponentially worse in terms of $|\mathcal{F}|$.

Can we do better then? The issue of this algorithm is that in the worst-case, every time we make a mistake, we can only eliminate one bad classifier. Ideally, we hope that a mistake will bring to us much more information, just like doing a binary search. It turns out that we can indeed do so by predicting the "majority vote" of the surviving classifiers, instead of just following any surviving one. This is the "Halving" algorithm, which at time t uses the following classifier:

$$\widehat{y}_t(x) = \mathrm{sign}\left(\sum_{f\in\mathcal{F}'} f(x)\right), \quad \text{where } \mathcal{F}' = \left\{f \in \mathcal{F} : \sum_{\tau=1}^{t-1} \mathbf{1}\{f(x_\tau) \neq y_\tau\} = 0\right\}.$$

Before analyzing the performance of this algorithm, note that this is an *improper* algorithm — $\widehat{y}_t$ is not from the class $\mathcal{F}$! Instead of writing down the improper function explicitly, it is often more convenient to equivalently change the learning protocol to: 1) environment first reveals x_t; 2) after seeing x_t, the learner makes a prediction -1 or $+1$ (in whatever way she likes); 3) environment reveals y_t. This setting is clearly more favorable to the learner if $\mathcal{D} = \mathcal{F}$, but it in fact does not give any extra power to an improper learner with $\mathcal{D} = \{-1, +1\}^{\mathcal{X}}$ since specifying how the learner predicts the label after seeing x_t is exactly the same as specifying an improper strategy from $\mathcal{D}$. For example, for the Halving algorithm, what the algorithm predicts at time t is simply the majority vote of the surviving classifiers on example x_t.

It is not hard too see that the Halving algorithm makes at most $\mathcal{O}(\log_2 |\mathcal{F}|)$ mistakes, since every time a mistake occurs, at least half of the classifiers are removed from the surviving set $\mathcal{F}'$. Consequently, the regret of the algorithm is also $\mathcal{O}(\log_2 |\mathcal{F}|)$, which is exponentially better than the naive strategy and is again n-independent. The realizable assumption is the key to getting a regret bound better than Equation (1).

2.2 General algorithm

So how can we go beyond the realizable setting, which is a rather strong assumption? First of all, we note that the two algorithms we discussed above are both *deterministic*, and on the other hand we argue that without the realizable assumption, no deterministic algorithm can guarantee sublinear regret, even for a very simple problem where there is only one possible example x and $\mathcal{F}$ contains only two constant functions $f_+(x) = +1$ and $f_-(x) = -1$. Indeed, because the algorithm is

deterministic, at the beginning of time t, its prediction on x is fixed and known to the environment already. Thus, if the environment always picks y_t to be the opposite of what the algorithm will predict, then the total loss of the learner is clearly just T, while one of f_- and f_+ must have total loss not larger than $T/2$, leading to linear regret at least $T/2$.

Recall that in Lecture 4, we briefly mentioned that the analogue of ERM or the so-called follow-the-leader strategy $\widehat{y}_t = \operatorname{argmin}_{f\in\mathcal{F}} \sum_{\tau=1}^{t-1} \ell(f, z_\tau)$ is not a good algorithm. The reason is clear now — this is a deterministic algorithm and will suffer linear regret even for trivial problems.

Therefore, we should shift our focus to randomized algorithms. Also, without the realizable assumption, it is clearly a bad idea to discard a classifier as soon as it makes a single mistake. Combining these two observations, one sees that a natural strategy is to randomly sample $f \in \mathcal{F}$ according to its previous performance — those with smaller cumulative loss so far are sampled with higher probability. Indeed, this leads to a classical algorithm that works for general problems (not just classification with 0-1 loss):

$$\textbf{Hedge}\text{: at time } t\text{, sample } \widehat{y}_t \in \mathcal{F} \text{ with probability } \Pr\left[\widehat{y}_t = f\right] \propto \exp\left(-\eta \sum_{\tau=1}^{t-1} \ell(f, z_\tau)\right) \tag{2}$$

where $\eta > 0$ is a parameter of the algorithm (called learning rate or step size). This simple algorithm turns out to be fundamental and has broad applications in many areas (we will discuss some in the future). In fact, it was (re)discovered in different areas and has many different names such as Hedge, multiplicative weight update, exponential weights, and so on.

Note that this is a proper algorithm. It can be viewed as doing a "soft" version of Follow-the-Leader because when η is infinity, the distribution simply puts all the mass on the current leader $\operatorname{argmin}_{f\in\mathcal{F}} \sum_{\tau=1}^{t-1} \ell(f, z_\tau)$ and the algorithm becomes Follow-the-Leader. In fact, this mapping (from cumulative losses to a exponential distribution) is also known as the "softmax" function.

We will show that Hedge ensures exactly regret bound (1), giving an algorithmic certification that finite classes are online learnable. We use the following lemma that will be useful for future discussions as well.

Lemma 1. *For any $\ell_t \in \mathbb{R}_+^K$ and $\eta > 0$, define $p_t \in \Delta(K)$ to be a distribution such that $p_t(i) \propto \exp\left(-\eta \sum_{\tau=1}^{t-1} \ell_\tau(i)\right)$, then we have for any $i^\star \in [K]$,*

$$\sum_{t=1}^{n} \langle p_t, \ell_t \rangle - \sum_{t=1}^{n} \ell_t(i^\star) \le \frac{\ln K}{\eta} + \eta \sum_{t=1}^{n} \sum_{i=1}^{K} p_t(i) \ell_t^2(i).$$

We defer the proof to the end of this section. With this lemma, it is straightforward to conclude the following.

Theorem 4. *For any environments (oblivious or adaptive), Hedge (defined in Equation (2)) with $\eta = \sqrt{\frac{\ln|\mathcal{F}|}{T}}$ ensures Equation (1).*

Proof. To apply Lemma 1, we set $K = |\mathcal{F}|$, rename the element of $\mathcal{F}$ by $1, \dots, K$, and set $\ell_t(i) = \ell(i, z_t)$, so that the learner exactly samples $\widehat{y}_t$ according to p_t as defined in Lemma 1. We then have

$$\mathbb{E}\left[\sum_{t=1}^{n} \ell(\widehat{y}_t, z_t)\right] = \mathbb{E}\left[\sum_{t=1}^{n} \langle p_t, \ell_t \rangle\right] \quad \text{and} \quad \mathbb{E}\left[\inf_{f\in\mathcal{F}} \sum_{t=1}^{n} \ell(f, z_t)\right] = \mathbb{E}\left[\min_{i^\star\in[K]} \sum_{t=1}^{n} \ell_t(i^\star)\right].$$

Applying Lemma 1 with expectation taken on both sides, using the fact $\ell_t(i) \le 1$, and plugging in the particular learning rate η (which is optimal) prove Equation (1). □

Next we prove Lemma 1 based on some potential-based approach. While the proof (or maybe even the algorithm Hedge itself) might seem to come out of nowhere, next time we will provide more explanation on this.

Proof of Lemma 1. Define $L_t = \sum_{\tau=1}^{t} \ell_\tau$ and "potential" $\Phi_t = \frac{1}{\eta} \ln\left(\sum_{i=1}^{K} \exp(-\eta L_t(i))\right)$. First we consider how the potential evolves from time $t-1$ to time t:

$$\begin{aligned}
\Phi_t - \Phi_{t-1} &= \frac{1}{\eta} \ln\left(\frac{\sum_{i=1}^{K} \exp(-\eta L_t(i))}{\sum_{i=1}^{K} \exp(-\eta L_{t-1}(i))}\right) \\
&= \frac{1}{\eta} \ln\left(\sum_{i=1}^{K} p_t(i) \exp(-\eta \ell_t(i))\right) \\
&\le \frac{1}{\eta} \ln\left(\sum_{i=1}^{K} p_t(i) \left(1 - \eta \ell_t(i) + \eta^2 \ell_t^2(i)\right)\right) && (e^{-y} \le 1 - y + y^2 \text{ for all } y \ge 0) \\
&= \frac{1}{\eta} \ln\left(1 - \eta \langle p_t, \ell_t \rangle + \eta^2 \sum_{i=1}^{K} p_t(i) \ell_t^2(i)\right) \\
&\le -\langle p_t, \ell_t \rangle + \eta \sum_{i=1}^{K} p_t(i) \ell_t^2(i). && (\ln(1+y) \le y)
\end{aligned}$$

Summing over t, telescoping and rearranging give

$$\begin{aligned}
\sum_{t=1}^{n} \langle p_t, \ell_t \rangle &\le \Phi_0 - \Phi_n + \eta \sum_{t=1}^{n} \sum_{i=1}^{K} p_t(i) \ell_t^2(i) \\
&\le \frac{\ln K}{\eta} - \frac{1}{\eta} \ln\left(\exp(-\eta L_n(i^\star))\right) + \eta \sum_{t=1}^{n} \sum_{i=1}^{K} p_t(i) \ell_t^2(i) \\
&\le \frac{\ln K}{\eta} + L_n(i^\star) + \eta \sum_{t=1}^{n} \sum_{i=1}^{K} p_t(i) \ell_t^2(i).
\end{aligned}$$

Further rearranging finishes the proof. □

3 Algorithms for Infinite Classes: Classification

We next consider learning infinite classes, starting from binary classification again. The goal is to construct an algorithm to learn any class with finite Littlestone dimension with regret bound

$$\mathbb{E}[\mathrm{Reg}(\mathcal{F}, n)] = \mathcal{O}\left(\sqrt{\mathrm{Ldim}(\mathcal{F}) n \ln n}\right). \tag{3}$$

Similar to the discussion for finite classes, we first make a realizable assumption that the class contains a perfect classifier. Note that the Halving algorithm does not work anymore since majority vote is not well-defined. However, the same idea of making a decision that brings most information still applies. Instead of halving the size of the class each time we make a mistake, for an infinite class the right analogue is to reduce the Littlestone dimension by one for each mistake we make. Specifically, consider the following algorithm that is again improper.

Let $\mathcal{F}' = \mathcal{F}$. For $t = 1, \ldots, n$,

1. receive x_t and define
$$\mathcal{F}'_- = \{f \in \mathcal{F}' \mid f(x_t) = -1\} \quad \text{and} \quad \mathcal{F}'_+ = \{f \in \mathcal{F}' \mid f(x_t) = +1\};$$
2. predict $\operatorname*{argmax}_{y \in \{-,+\}} \mathrm{Ldim}(\mathcal{F}'_y)$;
3. receive y_t and update $\mathcal{F}' \leftarrow \mathcal{F}'_{y_t}$.

Figure 1: Generalized Halving for Infinite Classes

In words, we split the current surviving class based on the prediction of a new example x_t and then follow the prediction of the subclass with larger Littlestone dimension. In the proof of the Sauer's lemma analogue in Lecture 5, we argued that one of $\mathcal{F}'_-$ and $\mathcal{F}'_+$ must have Littlestone dimension strictly smaller than $\mathcal{F}'$. Therefore, whenever we make a mistake, after the update of $\mathcal{F}'$ we must have reduced its Littlestone dimension by at least one. Therefore, it is clear that this algorithms makes at most $\mathrm{Ldim}(\mathcal{F})$ mistakes, which is also its regret.

This simple algorithm requires finding the Littlestone dimension though, and it might not be clear how to do so in general. However, for the simple function class $\mathcal{F} = \left\{ f_\theta(x) = \begin{cases} +1, & \text{if } x = \theta \\ -1, & \text{else} \end{cases} \middle| \, \theta \in \mathbb{R} \right\}$ we discussed last time, this becomes a naive algorithm that always predicts -1 until the first example x_t with label $+1$ appears, and afterwards always predicts according to f_{x_t}.

This generalization of Halving serves as an important building block for getting an algorithm with guarantee Equation (3) without the realizable assumption. To introduce the key idea of this algorithm, we first consider the following thought experiment. Note that an adaptive environment for a classification problem can be equivalently described by a pair of $\mathcal{X}$-valued tree $\boldsymbol{x}$ and $\{-1, +1\}$-valued tree $\boldsymbol{y}$, so that at time t, the environment selects $x_t = \boldsymbol{x}(\boldsymbol{s}_{1:t-1})$ and $y_t = \boldsymbol{y}(\boldsymbol{s}_{1:t-1})$, where $s_t \in \{-1, +1\}$ is the prediction of the learner for example x_t. Clearly x_t and y_t only depend on the decisions of learner prior to time t, which is exactly what an adaptive environment does.

Now, for a moment suppose we know the tree $\boldsymbol{x}$ (but not $\boldsymbol{y}$ so the problem does not become trivial). What we can do then is to construct a minimal zero-cover V of $\mathcal{F}|_{\boldsymbol{x}}$, and see each tree $\boldsymbol{v}$ in this cover as an "expert": at time t, this expert predicts $\boldsymbol{v}(\boldsymbol{s}_{1:t-1})$. Also note that since V is a cover, by definition for any $f \in \mathcal{F}$ and any sequence $\boldsymbol{s}$ of predictions of the learner that determines the sequence of $(x_1, y_1), \ldots, (x_n, y_n)$ chosen by the environment, there exists an expert $\boldsymbol{v}$ so that its prediction is identical to f on this sequence. Therefore, to ensure low regret against $\mathcal{F}$ is exactly the same as ensuring low regret to this pool of experts.

Since the number of experts is finite, we can just apply Hedge over this set of experts. Note that even though each expert is not exactly a classifier (i.e. a mapping from $\mathcal{X}$ to $\{-1, +1\}$) but rather an algorithm itself, the exact same analysis of Hedge still applies and we obtain a regret bound of order $\mathcal{O}\left(\sqrt{T \ln |V|}\right)$ by applying Lemma 1. Further applying Sauer's lemma analogue we know $\ln |V| = \mathcal{O}\left(\mathrm{Ldim}(\mathcal{F}) \ln n\right)$, which leads to a regret bound $\mathcal{O}\left(\sqrt{\mathrm{Ldim}(\mathcal{F}) n \ln n}\right)$, as in Equation (3).

The issue of this algorithm is of course that we do not know the tree $\boldsymbol{x}$ ahead of time. Also, constructing the exact cover seems very expensive as well (each tree has $2^n - 1$ nodes). However, a more careful examination reveals that this cover can in fact be constructed on-the-fly, so that we do not need to know $\boldsymbol{x}$ ahead of time and also do not need to explicitly construct each tree in the cover. Indeed, each tree can simply be "simulated" by a variant of the generalized Halving. We leave the details to HW2.