CSCI 678: Theoretical Machine Learning
Lecture 7

Fall 2024, Instructor: Haipeng Luo

1 Algorithms for Infinite Classes: Classification

In the last lecture, we discussed how to learn a finite class for binary classification problems using the simple Halving algorithm when the realizable assumption holds and more generally using the Hedge algorithm without making the realizable assumption. Continuing that discussion, we next consider learning an infinite classes of binary classifiers. The goal is to construct an algorithm to learn any class with a finite Littlestone dimension $\text{Ldim}(\mathcal{F}) = d$ and ensure the following:

$$\mathbb{E}[\text{Reg}(\mathcal{F}, n)] = \mathcal{O}\left(\sqrt{dn \ln n}\right). \tag{1}$$

We will only discuss improper algorithms this time, in which case, as discussed last time, the learning protocol is equivalent to: in each round t, 1) environment first reveals $x_t \in \mathcal{X}$; 2) after seeing x_t, the learner makes a prediction -1 or $+1$; 3) finally, the environment reveals $y_t \in \{-1, +1\}$.

Warm-up. Similar to the discussion for finite classes, we first make a realizable assumption: 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.

Algorithm 1: Generalized Halving for Infinite Classes

```
Let $\mathcal{F}' = \mathcal{F}$. For $t = 1, \ldots, n$,
    1. Receive $x_t$ and define
           $\mathcal{F}'_- = \{f \in \mathcal{F}' \mid f(x_t) = -1\}$  and  $\mathcal{F}'_+ = \{f \in \mathcal{F}' \mid f(x_t) = +1\}$.
    2. Predict $\operatorname{argmax}_{y \in \{-,+\}} \text{Ldim}(\mathcal{F}'_y)$.
    3. Receive $y_t$ and update $\mathcal{F}' \leftarrow \mathcal{F}'_{y_t}$.
```

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 a 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, leading to the following guarantee.

Theorem 1. *Under the realizable assumption, Generalized Halving (Algorithm 1) makes at most* $\text{Ldim}(\mathcal{F})$ *mistakes (which is also its regret).*

This simple algorithm requires actually computing 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}. (You are encouraged to think about how this algorithm behaves for the high-dimensional generalization of this simple class studied in HW2.)

Lifting the realizable assumption. This generalization of Halving serves as an important building block for developing an algorithm with guarantee Equation (1) without the realizable assumption. To introduce the key idea of this algorithm, we first consider the following thought experiment. Note that an adaptive (and deterministic) 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 the 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 $\boldsymbol{v}(\boldsymbol{s}_{1:t-1})$. Note that since V is a cover, by definition, for any $f \in \mathcal{F}$ and any sequence $\boldsymbol{s}$ of the learner's predictions, there exists an expert $\boldsymbol{v}$ so that its prediction on the path determined by $\boldsymbol{s}$ is identical to f. Therefore, to ensure low regret against $\mathcal{F}$ is exactly the same as ensuring low regret against 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 (that is, not a mapping from $\mathcal{X}$ to $\{-1,+1\}$), the exact same analysis of Hedge still applies, leading to a regret bound of order $\mathcal{O}\left(\sqrt{n \ln |V|}\right)$ according to Lemma 1 from the last lecture. Further applying Sauer's lemma analogue, we know $\ln |V| = \mathcal{O}(d \ln n)$, and thus the regret is of order $\mathcal{O}\left(\sqrt{dn \ln n}\right)$, as in Equation (1).

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 (each tree has $2^n - 1$ nodes). On the other hand, note that in this algorithm, the zero-cover V intuitively contains a lot of redundant information, since at the end we only care about one particular path of each tree in V. Can we make use of this fact to construct each tree in the cover partially (that is, only focus on the path that matters) and adaptively (that is, decide what level t should be only after seeing the example x_t in round t?)

Constructing the cover partially and adaptively. The answer turns out to be yes. To introduce this method, we first define (but not explicitly construct) the following zero-cover V of $\mathcal{F}|\boldsymbol{x}$. Each element in V, denoted by $v^{\mathcal{T}}$, is indexed by $\mathcal{T}$, which is a set of m time steps $1 \le t_1 < t_2 < \cdots < t_m \le n$ for some integer $m \le d$. This implies that V has exactly $\sum_{m=0}^{d} \binom{n}{m}$ elements. For a fixed $\mathcal{T}$, the value of $v^{\mathcal{T}}$ is defined in a top-down manner. Specifically, for any path $\boldsymbol{\epsilon}$ and any level t, the corresponding node $v_t^{\mathcal{T}}(\boldsymbol{\epsilon})$ is defined as follows:

- let $\mathcal{F}' = \{f \in \mathcal{F} \mid f(\boldsymbol{x}_s(\boldsymbol{\epsilon})) = v_s^{\mathcal{T}}(\boldsymbol{\epsilon}) \text{ for all } s = 1, \ldots, t-1\}$ be the subclass of "surviving" classifiers that agree with the values of $v^{\mathcal{T}}$ on the path $\boldsymbol{\epsilon}$ before level t;
- split the surviving classifiers into two groups based on their prediction on the example $\boldsymbol{x}_t(\boldsymbol{\epsilon})$: $\mathcal{F}'_- = \{f \in \mathcal{F}' \mid f(\boldsymbol{x}_t(\boldsymbol{\epsilon})) = -1\}$ and $\mathcal{F}'_+ = \{f \in \mathcal{F}' \mid f(\boldsymbol{x}_t(\boldsymbol{\epsilon})) = +1\}$;
- if $t \notin \mathcal{T}$, define $v_t^{\mathcal{T}}(\boldsymbol{\epsilon})$ as $\operatorname{argmax}_{y \in \{-,+\}} \operatorname{Ldim}(\mathcal{F}'_y)$; otherwise, define it as the opposite label.

You might already notice that the definition above shares some common elements with the generalized Halving algorithm. Indeed, we will utilize this similarity to prove that V is indeed a zero-cover.

Lemma 1. *The set V defined above is a zero-cover for $\mathcal{F}|_{\boldsymbol{x}}$.*

Proof. For any f and $\boldsymbol{\epsilon} \in \{-1,+1\}^n$, imagine running generalized Halving on the sequence $(\boldsymbol{x}_1(\boldsymbol{\epsilon}), f(\boldsymbol{x}_1(\boldsymbol{\epsilon}))), \ldots, (\boldsymbol{x}_n(\boldsymbol{\epsilon}), f(\boldsymbol{x}_n(\boldsymbol{\epsilon})))$ and let m be the number of mistakes it makes in this process. Since the realizable assumption holds by construction, we know $m \le d$ according to Theorem 1. Let $\mathcal{T} = \{t_1, \ldots, t_m\}$ be the time steps where it makes these m mistakes. We claim that $v^{\mathcal{T}}$ is such that $v_t^{\mathcal{T}}(\boldsymbol{\epsilon}) = f(\boldsymbol{x}_t(\boldsymbol{\epsilon}))$ for all $t = 1, \ldots, n$, certifying that V is indeed a zero-cover of $\mathcal{F}|_{\boldsymbol{x}}$. Indeed, on the path $\boldsymbol{\epsilon}$, by the definition of $v^{\mathcal{T}}$, its value at the root is the same as what Halving predicts if it predicts correctly, and the opposite of what it predicts if it errs, meaning $v_1^{\mathcal{T}}(\boldsymbol{\epsilon}) = f(\boldsymbol{x}_1(\boldsymbol{\epsilon}))$

always. In either case, the surviving subclass updated by Halving is the same as that in the definition of $v_2^{\mathcal{T}}$, which then by the same reasoning implies $v_2^{\mathcal{T}}(\boldsymbol{\epsilon}) = f(\boldsymbol{x}_2(\boldsymbol{\epsilon}))$ as well. Repeating this argument finishes the proof. $\square$

Note that this lemma also gives a different proof for the Sauer's lemma analogue, that is, $\mathcal{N}_0(\mathcal{F}|_{\boldsymbol{x}}) \le \sum_{m=0}^{d} \binom{n}{m}$. While the proof we discussed last time also provides an explicit way to recursively construct a zero-cover, the advantage of the new method discussed here is that it in fact allows us to construct a zero-cover partially and adaptively. Indeed, from the definition of $v^{\mathcal{T}}$, we can see that if we only care about one particular path $\boldsymbol{\epsilon}$ on the tree, which corresponds to the actual sequence of examples $x_1, \dots, x_n$ chosen by the environment, we can obtain the value of each $v_t^{\mathcal{T}}(\boldsymbol{\epsilon})$ *just in time*, that is, right after seeing x_t, via the following procedure.

Algorithm 2: An expert indexed by $\mathcal{T}$ that outputs one path of $v^{\mathcal{T}}$ on the fly

```
Let $\mathcal{F}' = \mathcal{F}$. For $t = 1, \dots, n$,
    1. Receive $x_t$ and define
           $\mathcal{F}'_- = \{f \in \mathcal{F}' \mid f(x_t) = -1\}$   and   $\mathcal{F}'_+ = \{f \in \mathcal{F}' \mid f(x_t) = +1\}$.
    2. If $t \notin \mathcal{T}$, predict $\mathrm{argmax}_{y \in \{-,+\}}$ Ldim$(\mathcal{F}'_y)$; otherwise, predict the opposite label. Let
       $y_t^{\mathcal{T}}$ be this prediction.
    3. Update $\mathcal{F}' \leftarrow \mathcal{F}'_{y_t^{\mathcal{T}}}$.
```

In other words, we can see this procedure as an expert (indexed by $\mathcal{T}$) that follows the same learning protocol: in each round t, predicts $y_t^{\mathcal{T}}$ after seeing x_t. By definition, its n outputs exactly correspond to the value of $v^{\mathcal{T}}$ on the path that we actually care about, and it does so without any knowledge of the rest of $\boldsymbol{x}$. This expert can also be seen as a variant of the Generalized Halving algorithm, with two differences: first, it deviates from Halving (by predicting the opposite) for all $t \in \mathcal{T}$; and second, when updating the surviving class, it is "self-confident" and always treats its prediction $y_t^{\mathcal{T}}$ as the "correct" label.

Given these experts, our final algorithm is the same as before: use Hedge to pick an expert in each round and follow its prediction. This is summarized below:

Algorithm 3: Hedge over partial covers

```
Parameter: learning rate $\eta > 0$.
For each possible $\mathcal{T}$ that is a set of $m$ time steps $1 \le t_1 < t_2 < \cdots < t_m \le n$ for some integer
  $m \le d = $ Ldim$(\mathcal{F})$, maintain an expert indexed by $\mathcal{T}$ as defined in Algorithm 2.
For $t = 1, \dots, n$,
    1. Receive $x_t$ and send it to all experts.
    2. Sample an expert $\mathcal{T}_t$ according to the distribution:
           $\Pr[\mathcal{T}_t = \mathcal{T}] \propto \exp\left(-\eta \sum_{s=1}^{t-1} \mathbf{1}\left\{y_s^{\mathcal{T}} \neq y_s\right\}\right)$
       and follow its prediction $y_t^{\mathcal{T}_t}$.
    3. Receive the true label $y_t$.
```

The following result is then a direct application of Hedge's regret guarantee.

Theorem 2. *Algorithm 3 ensures* $\mathbb{E}[\mathrm{Reg}(\mathcal{F}, n)] = \mathcal{O}\left(\sqrt{dn \ln n}\right)$.

Proof. By Lemma 1 and the fact that expert indexed by $\mathcal{T}$ simulates the tree $v^{\mathcal{T}}$, the regret of Algorithm 3 against $\mathcal{F}$ is the same as its regret against the best expert. Since the number of experts is $\sum_{m=0}^{d} \binom{n}{m} \le \left(\frac{en}{d}\right)^d$ (proven in Sauer's lemma), applying Hedge's regret guarantee for a finite class directly proves the theorem. $\square$

To conclude, we have exactly achieved our goal stated in Equation (1) via an improper algorithm. We mention in passing that achieving the same with a proper algorithm was an open question for about a decade, but recently resolved by Hanneke et al. [2021].

2 Classification with Margin

While achieving Equation (1) algorithmically is theoretically interesting, it is practically quite limited since 1) the running time of Algorithm 3 is exponential in d and 2) as we discussed, classes with a finite Littlestone dimension are very restricted. To address this issue, we discussed in the last lecture that in practice we often consider some surrogate of the 0-1 loss, making the problem closer to a regression task. We start with one simple example in this section, focusing on learning a linear class $\mathcal{F} = \left\{f_\theta(x) = \langle \theta, x \rangle \mid \theta \in B_p^d\right\}$ with the hinge loss $\ell(f,(x,y)) = \max\{1 - yf(x), 0\}$.

As the first step, we again make a realizable assumption: $\inf_{f\in\mathcal{F}} \sum_{t=1}^n \max\{1 - y_t f(x_t), 0\} = 0$, which is equivalent to saying that there exists $\theta^\star \in B_p^d$ such that $y_t \langle \theta^\star, x_t \rangle \geq 1$ holds for all $t = 1, \ldots, n$. Note that this is an even stronger assumption compared to the realizable assumption with respect to 0-1 loss, and it posits that the data is not only linearly separable by some hyperplane, but is separable with margin at least 1. In fact, a more standard form of this assumption is to normalize the data x_t, leading to a different margin parameter:

Assumption 1 (γ-margin assumption)**.** *Data is normalized such that $x_t \in B_q^d$, and there exists a constant $\gamma > 0$ and a hyperplane parameterized by $\theta^\star \in B_p^d$ (for some $p, q \geq$ with $\frac{1}{p} + \frac{1}{q} = 1$) such that $y_t \langle \theta^\star, x_t \rangle \geq \gamma$ holds for all $t = 1, \ldots, n$.*

Under this margin assumption, one trivial but inefficient approach is to construct a pointwise $\gamma/2$-cover of $\mathcal{F}$ with size $\mathcal{N}(\mathcal{F}, \alpha) \leq (\frac{4}{\gamma} + 1)^d$ (Proposition 2 of Lecture 3), and then run Halving over this finite cover. Indeed, by the covering property, there exists θ' that is the “representative” of $\theta^\star$ and such that

$$y_t \langle \theta', x_t \rangle = y_t \langle \theta^\star, x_t \rangle + y_t \langle \theta' - \theta^\star, x_t \rangle \geq \gamma - \gamma/2 > 0,$$

which means that the realizable assumption with 0-1 loss holds for this finite cover and thus Halving makes at most

$$\mathcal{O}\left(\ln \mathcal{N}(\mathcal{F}, \alpha)\right) = \mathcal{O}\left(d \ln\left(\frac{4}{\gamma} + 1\right)\right) \tag{2}$$

mistakes.

How do we obtain a more efficient algorithm? In the following, we focus on the case $p = q = 2$ (see HW3 for the case $p = 1$ and $q = \infty$). In this case, the margin condition $y_t \langle \theta^\star, x_t \rangle \geq \gamma$ implies that the Euclidean distance of each data point x_t is at least γ away from the hyperplane $\theta^\star$. Below, we describe a simple algorithm called *Perceptron*.

Algorithm 4: Perceptron Algorithm

```
Let $\theta = \mathbf{0}$. For $t = 1, \ldots, n$:
    1. Receive $x_t$ and predict $s_t = \text{sign}(\langle x_t, \theta \rangle)$.
    2. Receive $y_t$. If $y_t \neq s_t$, update $\theta \leftarrow \theta + y_t x_t$.
```

Note that Perceptron is extremely efficient and it updates itself (the weight vector θ) if and only if it makes a mistake. The update is simply to add the current misclassified example x_t to θ with the correct direction (determined by y_t), so that the corresponding hyperplane rotates towards a direction that corrects the previous mistake to some degree. Indeed, whenever a mistake is made, that is $y_t \langle x_t, \theta \rangle \leq 0$, immediately after the update the algorithm is more likely to be correct on x_t since $y_t \langle x_t, \theta + y_t x_t \rangle = y_t \langle x_t, \theta \rangle + \|x_t\|_2^2$ and $\|x_t\|_2^2 \geq 0$. Also note that this is a deterministic and improper algorithm (θ might not in B_p^d.).

Perceptron is guaranteed to make no more than a constant number of mistakes under the margin assumption, as shown in the following theorem.

Theorem 3. *Suppose that the γ-margin assumption holds with $p = q = 2$. Then Perceptron makes at most $1/\gamma^2$ mistakes.*

Proof. Denote the weight vector maintained by the algorithm at the beginning of round t as θ_t, which means $\theta_1 = \mathbf{0}$ and $\theta_{t+1} = \theta_t + \mathbf{1}\{s_t \neq y_t\} y_t x_t$. We first show that the correlation between $\theta^\star$ and θ_t is non-decreasing:

$$\langle \theta_{t+1}, \theta^\star \rangle = \langle \theta_t + \mathbf{1}\{s_t \neq y_t\} y_t x_t, \theta^\star \rangle \geq \langle \theta_t, \theta^\star \rangle + \mathbf{1}\{s_t \neq y_t\} \gamma,$$

where the last step uses the γ-margin assumption. With $M = \sum_{t=1}^n \mathbf{1}\{s_t \neq y_t\}$ being the total number of mistakes we thus have $M\gamma \leq \langle \theta_{T+1}, \theta^\star \rangle \leq \|\theta_{T+1}\|_2$. Next, we show that the norm of θ_{T+1} cannot be too large since

$$\begin{aligned}
\|\theta_{t+1}\|_2^2 &= \|\theta_t + \mathbf{1}\{s_t \neq y_t\} y_t x_t\|_2^2 \\
&= \|\theta_t\|_2^2 + 2\mathbf{1}\{s_t \neq y_t\} \langle \theta_t, y_t x_t \rangle + \mathbf{1}\{s_t \neq y_t\} \|x_t\|_2^2 \\
&\leq \|\theta_t\|_2^2 + \mathbf{1}\{s_t \neq y_t\}
\end{aligned}$$

and thus $\|\theta_{T+1}\|_2 \leq \sqrt{M}$. Combining these two facts gives $M \leq 1/\gamma^2$. □

Even though the mistake bound $1/\gamma^2$ has a worse dependence on γ compared to Equation (2), it is on the other hand completely *dimension free*, making the algorithm especially preferable for problems with a huge dimension. Note that this phenomenon (of having no dimension dependence but worse dependence on γ) is exactly the same as what we saw in Section 2.1 of Lecture 4, where we show an almost dimension-independent log covering number of order $1/\gamma^2$ for the linear class. In fact, using the guarantee of Perceptron, one can prove that the sequential fat-shattering dimension of this linear class is exactly of order $1/\gamma^2$ (see HW3).

3 Online Convex Optimization

How do we learn in general without the margin assumption? To introduce an efficient solution, we come back to the general setup where at each time the learner selects $\widehat{y}_t \in \mathcal{F}$ (for simplicity we now consider proper learners) and the environment decides $z_t \in \mathcal{Z}$. Instead of discussing how to learn a general class with a finite sequential fat-shattering dimension, which does not generally admit an efficient algorithm, we will instead consider the case where $\mathcal{F}$ is a convex set and the loss function $\ell(\cdot, z)$ is convex in the first argument for any $z \in \mathcal{Z}$. This is known as the *Online Convex Optimization* (OCO) framework.

Many problems fall into this framework or can be re-parameterized to fit into this framework. For instance, in the previous example of learning a linear class with hinge loss, one can equivalently see the decision set as $\mathcal{F} = B_p^d$ and the loss function becomes $\ell(f, (x, y)) = \max\{1 - y\langle f, x \rangle, 0\}$, both of which are convex. Learning a linear class with other common losses (such as logistic loss or square loss) is the same story. For the finite class example we studied last time, while in the natural representation $\mathcal{F}$ is a discrete finite set (which is of course not convex), one can instead take $\mathcal{F}'$ to be the simplex of distributions over the finite elements of $\mathcal{F}$, which is convex, and take the expected loss $\mathbb{E}_{f \sim f'}[\ell(f, z)]$ as the new loss function, which is linear (and thus convex) in f'.

We first point out that the case when $\ell(f, z) = \langle f, z \rangle$ is a linear function is in some sense universal. Indeed, by convexity, we can upper bound the regret in the general case as

$$\sum_{t=1}^n \ell(\widehat{y}_t, z_t) - \sum_{t=1}^n \ell(f^\star, z_t) \leq \sum_{t=1}^n \langle \widehat{y}_t - f^\star, \nabla \ell(\widehat{y}_t, z_t) \rangle, \tag{3}$$

which becomes the regret for a problem with linear loss function $\langle f, \nabla \ell(\widehat{y}_t, z_t) \rangle$ at time t. Even though the loss function now actually depends on the decision of the learner, it turns out that this is not a problem in this formulation as we will see soon. This reduction ignores the curvature of the original convex loss functions and might not lead to the optimal solutions. For simplicity, however, we will mainly focus on linear loss functions, denoted as $\ell(f, z) = \langle f, z \rangle$.

There are several general and efficient approaches to solve this problem. Here, we discuss one of them called *Follow-the-Regularized-Leader* (FTRL), defined as

$$\text{FTRL:} \quad \widehat{y}_t = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{\tau=1}^{t-1} \langle f, z_\tau \rangle + \frac{1}{\eta} \psi(f)$$

where $\eta > 0$ is some learning rate and $\psi : \mathcal{F} \to \mathbb{R}$ is some *regularizer* that penalizes the learner for making a decision too close to that of Follow-the-Leader (FTL) (indeed, without the regularization term this is just FTL). We require that the regularizer is 1-strongly convex with respect to some norm $\|\cdot\|$, that is, for any $f, f' \in \mathcal{F}$:

$$\psi(f) \leq \psi(f') + \langle \nabla\psi(f), f - f' \rangle - \frac{1}{2} \|f - f'\|^2 . \tag{4}$$

Strong convexity ensures that $\widehat{y}_t$ is unique. Moreover, as we will see in the analysis, strong convexity also ensures *stability* of the algorithm, which turns out to be essential to ensure small regret. Last but not least, (strong) convexity of the regularizer also ensures that the optimization required by FTRL can be efficiently solved by any convex optimization algorithms. Before diving into the analysis, we first consider two canonical examples.

Recovering Gradient Descent. Consider $\psi(f) = \frac{1}{2}\|f\|_2^2$, which is strongly convex with respect to the ℓ_2 norm (in fact, Equation (4) holds with equality). In this case, FTRL becomes

$$\widehat{y}_t = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{\tau=1}^{t-1} \langle f, z_\tau \rangle + \frac{1}{2\eta}\|f\|_2^2 = \operatorname*{argmin}_{f \in \mathcal{F}} \left\| f + \eta \sum_{\tau=1}^{t-1} z_\tau \right\|_2^2 .$$

If we let $\widehat{y}'_t = \widehat{y}'_{t-1} - \eta z_t$, then $\widehat{y}_{t+1} = \operatorname{argmin}_{f \in \mathcal{F}} \|f - \widehat{y}'_t\|_2^2$. Note that z_t corresponds to the gradient of the loss function at $\widehat{y}_t$ in the reduction of Equation (3). Therefore, this is exactly the (lazy version) of projected gradient descent.

Recovering Hedge. As mentioned earlier, to capture the finite class case we can take $\mathcal{F}$ to be a simplex. In this case consider taking $\psi(f) = \sum_i f(i) \ln f(i)$ to be the (negative) entropy, which is strongly convex with respect to the ℓ_1 norm. Indeed, it is not hard to verify that Equation (4) is equivalent to the well-known Pinsker's inequality: $\frac{1}{2}\|f - f'\|_1^2 \leq \mathrm{KL}(f', f)$ (proof omitted). Applying KKT conditions, it is also straightforward to see that the solution of FTRL is exactly $\widehat{y}_t(i) \propto \exp(-\eta \sum_{\tau=1}^{t-1} z_\tau(i))$, which is simply Hedge.

There are many other algorithms that can be formulated as FTRL. The general regret guarantee for FTRL is shown in the following theorem.

Theorem 4. *FTRL with learning rate η and a 1-strongly-convex regularizer ψ (with respect to some norm $\|\cdot\|$) ensures*

$$\mathrm{Reg}(\mathcal{F}, n) = \max_{f^\star \in \mathcal{F}} \sum_{t=1}^{n} \langle \widehat{y}_t - f^\star, z_t \rangle \leq \frac{R}{\eta} + \eta \sum_{t=1}^{n} \|z_t\|_\star^2 ,$$

where $R = \max_{f \in \mathcal{F}} \psi(f) - \min_{f \in \mathcal{F}} \psi(f)$ is the range of the regularizer and $\|\cdot\|_\star$ is the dual norm of $\|\cdot\|$. If we further have $\|z_t\|_\star \leq G$ for all t, then picking $\eta = \sqrt{\frac{R}{nG^2}}$ gives $\mathrm{Reg}(\mathcal{F}, n) \leq 2G\sqrt{Rn}$.

Again, according to the reduction of Equation (3), the condition $\|z_t\|_\star \leq G$ exactly corresponds to a Lipschitz condition of the loss function. This also provides a guidance on how to choose the regularizer ψ — if the loss function has a small Lipschitz condition with respect to some norm $\|\cdot\|_\star$, then we should choose a regularizer that is strongly convex with respect to the dual norm $\|\cdot\|$ to exploit this fact. For example, if the loss function is Lipschitz in ℓ_2 norm, then we can choose $\psi(f) = \frac{1}{2}\|f\|_2^2$ and apply gradient descent. On the other hand, if the loss function is Lipschitz in ℓ_∞ norm and the decision space $\mathcal{F}$ is the simplex over a finite set of size K, then we can choose ψ to be the (negative) entropy and apply Hedge. Note that in this case, $R = \max_{f \in \mathcal{F}} \psi(f) - \min_{f \in \mathcal{F}} \psi(f) \leq \ln K$, and Theorem 4 recovers the Hedge regret bound $\mathcal{O}(\sqrt{n \ln K})$ we proved last time.

3.1 Proof of Theorem 4

We make use of two important lemmas. The first one analyzes the regret of an imaginary strategy called Be-the-Regularized-Leader (BTRL), which predicts $\widehat{y}_{t+1}$ at time t. This is of course not a valid algorithm since $\widehat{y}_{t+1}$ depends on z_t. However, understanding the regret of this imaginary strategy turns out to be very useful.

Lemma 2 (Be-the-Regularized-Leader Lemma)**.** *FTRL with learning rate η ensures for any $f^\star \in \mathcal{F}$,*

$$\sum_{t=1}^{n} \langle \widehat{y}_{t+1} - f^\star, z_t \rangle \le \frac{R}{\eta}$$

Proof. Let $h_t(f) = \langle f, z_t \rangle$ for $t = 1, \ldots, T$ and $h_0(f) = \frac{1}{\eta}\psi(f)$. Then FTRL strategy is $\widehat{y}_t = \operatorname{argmin}_{f\in\mathcal{F}} \sum_{\tau=0}^{t-1} h_\tau(f)$. We then have

$$\begin{aligned}
\sum_{t=0}^{n} h_t(\widehat{y}_{t+1}) - h_t(f^\star) &\le \sum_{t=0}^{n} h_t(\widehat{y}_{t+1}) - h_t(\widehat{y}_{n+1}) && \text{(By the optimality of } \widehat{y}_{n+1}\text{)}\\
&= \sum_{t=0}^{n-1} h_t(\widehat{y}_{t+1}) - h_t(\widehat{y}_{n+1}) \le \sum_{t=0}^{n-1} h_t(\widehat{y}_{t+1}) - h_t(\widehat{y}_{n}) && \text{(By the optimality of } \widehat{y}_{n}\text{)}\\
&= \sum_{t=0}^{n-2} h_t(\widehat{y}_{t+1}) - h_t(\widehat{y}_{n}) \le \cdots \le 0.
\end{aligned}$$

Rearranging shows

$$\sum_{t=1}^{n} \langle \widehat{y}_{t+1} - f^\star, z_t \rangle = \sum_{t=1}^{n} h_t(\widehat{y}_{t+1}) - h_t(f^\star) \le h_0(f^\star) - h_0(\widehat{y}_1) = \frac{\psi(f^\star) - \min_{f\in\mathcal{F}} \psi(f)}{\eta} \le \frac{R}{\eta}.$$

□

With this lemma, bounding the regret of FTRL simply boils down to analyzing the stability of the algorithm:

$$\sum_{t=1}^{n} \langle \widehat{y}_t - f^\star, z_t \rangle = \sum_{t=1}^{n} \langle \widehat{y}_{t+1} - f^\star, z_t \rangle + \sum_{t=1}^{n} \langle \widehat{y}_t - \widehat{y}_{t+1}, z_t \rangle \le \frac{R}{\eta} + \sum_{t=1}^{n} \|\widehat{y}_t - \widehat{y}_{t+1}\| \, \|z_t\|_\star . \quad (5)$$

The next lemma then shows that FTRL is indeed stable thanks to the strong convexity of the regularizer.

Lemma 3. *FTRL with learning rate η and a 1-strongly-convex regularizer ψ (with respect to norm $\|\cdot\|$) ensures $\|\widehat{y}_t - \widehat{y}_{t+1}\| \le \eta \|z_t\|_\star$ for all $t = 1, \ldots, T$.*

Proof. Let $H_t(f) = \sum_{\tau=1}^{t-1} \langle f, z_\tau \rangle + \frac{1}{\eta}\psi(f)$ so that $\widehat{y}_t = \operatorname{argmin}_{f\in\mathcal{F}} H_t(f)$ and $\widehat{y}_{t+1} = \operatorname{argmin}_{f\in\mathcal{F}} H_{t+1}(f)$. By strong convexity and first order optimality, we have

$$H_t(\widehat{y}_t) \le H_t(\widehat{y}_{t+1}) + \langle \nabla H_t(\widehat{y}_t), \widehat{y}_t - \widehat{y}_{t+1} \rangle - \frac{1}{2\eta} \|\widehat{y}_t - \widehat{y}_{t+1}\|^2 \le H_t(\widehat{y}_{t+1}) - \frac{1}{2\eta} \|\widehat{y}_t - \widehat{y}_{t+1}\|^2 .$$

By the same reasoning, we also have

$$H_{t+1}(\widehat{y}_{t+1}) \le H_{t+1}(\widehat{y}_t) + \langle \nabla H_{t+1}(\widehat{y}_{t+1}), \widehat{y}_{t+1} - \widehat{y}_t \rangle - \frac{1}{2\eta} \|\widehat{y}_{t+1} - \widehat{y}_t\|^2 \le H_{t+1}(\widehat{y}_t) - \frac{1}{2\eta} \|\widehat{y}_{t+1} - \widehat{y}_t\|^2 .$$

Combining and rearranging give

$$\begin{aligned}
\|\widehat{y}_t - \widehat{y}_{t+1}\|^2 &\le \eta(H_t(\widehat{y}_{t+1}) - H_{t+1}(\widehat{y}_{t+1}) + H_{t+1}(\widehat{y}_t) - H_t(\widehat{y}_t))\\
&= \eta(\langle \widehat{y}_t, z_t \rangle - \langle \widehat{y}_{t+1}, z_t \rangle)\\
&= \eta \langle \widehat{y}_t - \widehat{y}_{t+1}, z_t \rangle \le \eta \|\widehat{y}_t - \widehat{y}_{t+1}\| \, \|z_t\|_\star .
\end{aligned}$$

Further dividing both sides by $\|\widehat{y}_t - \widehat{y}_{t+1}\|$ finishes the proof. □

Combining this lemma with Equation (5) proves Theorem 4. Also note that the fact that z_t might depend on $\widehat{y}_t$ (which is indeed the case in the reduction of Equation (3)) does not affect the result, as mentioned earlier.

References

Steve Hanneke, Roi Livni, and Shay Moran. Online learning with simple predictors and a combinatorial characterization of minimax in 0/1 games. In *Conference on Learning Theory*, pages 2289–2314. PMLR, 2021.