Theoretical Machine Learning
Lecture 7

Instructor: Haipeng Luo

1 Classification with Margin

In the last lecture, we discussed how to learn a binary class with finite Littlestone dimension. However, these classes are quite restricted and the results are very limited. Based on previous discussions, we will instead consider minimizing surrogate of 0-1 loss. For example, consider learning linear class $\mathcal{F} = \left\{ f_\theta(x) = \langle \theta, x \rangle \mid \theta \in B_p^d \right\}$ with hinge loss $\ell(f, (x, y)) = \max\{1 - yf(x), 0\}$.

As the first step, we again make a realizable assumption: $\inf_{f \in \mathcal{F}} \sum_{t=1}^{T} \max\{1 - y_t f(x_t), 0\} = 0$, 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 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 $\mathcal{N}(\mathcal{F}, \alpha) \leq (\frac{4}{\gamma} + 1)^d$, 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

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

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*, which in some case can be seen as the starting point of neural networks.

```
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 $\theta \leftarrow \theta + y_t x_t$.
```

Figure 1: Perceptron Algorithm

Note that Perceptron in 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.

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

Theorem 1. *Suppose 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 (1), it is on the other hand completely *dimension free*, making the algorithm especially preferable for problems with a huge dimension.

2 Online Convex Optimization

How do we learn in general without the margin assumption? To introduce a solution, we come back to the general setup where at each time the learner selects $\widehat{y}_t \in \mathcal{F}$ (for simplicity we consider proper learners) and the environment decides $z_t \in \mathcal{Z}$. The only assumption we will make is that $\mathcal{F}$ 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 also 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

$$\sum_{t=1}^n \ell(\widehat{y}_t, z_t) - \sum_{t=1}^n \ell(f^\star, z_t) \leq \sum_{t=1}^n \langle \widehat{y}_t - f^\star, \nabla \ell(\widehat{y}_t, z_t) \rangle, \tag{2}$$

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 $f, f' \in \mathcal{F}$:

$$\psi(f) \le \psi(f') + \langle \nabla\psi(f), f - f'\rangle - \frac{1}{2}\left\|f - f'\right\|^2. \tag{3}$$

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 to the analysis, we first consider two classic 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 (3) 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 (2). 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 (3) is equivalent to the well-known Pinsker's inequality: $\frac{1}{2}\|f - f'\|_1^2 \le \mathrm{KL}(f', f)$ (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 2. *FTRL with learning rate η and a 1-strongly-convex regularizer ψ (with respect to some norm $\|\cdot\|$) ensures*

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

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 $\|z_t\|_\star \le G$ for all t, then picking $\eta = \sqrt{\frac{R}{nG^2}}$ gives $\mathrm{Reg}(\mathcal{F}, n) \le G\sqrt{Rn}$.

Again, according to the reduction of Equation (2), the condition $\|z_t\|_\star \le G$ 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, then we can choose ψ to be the (negative) entropy and apply Hedge. Note that in this case, $R = \max_{f\in\mathcal{F}}\psi(f) - \min_{f\in\mathcal{F}}\psi(f) \le \ln|\mathcal{F}|$, and Theorem 2 recovers the Hedge regret bound $\mathcal{O}(\sqrt{n\ln|\mathcal{F}|})$ we proved last time.

2.1 Proof of Theorem 2

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 1 (Be-the-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

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

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:

$$\sum_{t=1}^{n} \langle \widehat{y}_t - f^\star, z_t \rangle = \sum_{t=1}^{n} \langle \widehat{y}_{t+1} - f^\star, z_t \rangle + \sum_{t=1}^{n} \langle \widehat{y}_t - \widehat{y}_{t+1}, z_t \rangle \le \frac{R}{\eta} + \sum_{t=1}^{n} \|\widehat{y}_t - \widehat{y}_{t+1}\| \, \|z_t\|_\star . \quad (4)$$

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

Lemma 2. *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 $H(f) = \sum_{\tau=1}^{t-1} \langle f, z_\tau \rangle + \frac{1}{\eta}\psi(f)$ and $H'(f) = \sum_{\tau=1}^{t} \langle f, z_\tau \rangle + \frac{1}{\eta}\psi(f)$ so that $\widehat{y}_t = \operatorname{argmin}_{f\in\mathcal{F}} H(f)$ and $\widehat{y}_{t+1} = \operatorname{argmin}_{f\in\mathcal{F}} H'(f)$. By strong convexity and first order optimality, we have

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

By the same reasoning, we also have

$$H'(\widehat{y}_{t+1}) \le H'(\widehat{y}_t) + \langle \nabla H'(\widehat{y}_{t+1}), \widehat{y}_{t+1} - \widehat{y}_t \rangle - \frac{1}{2\eta}\|\widehat{y}_{t+1} - \widehat{y}_t\|^2 \le H'(\widehat{y}_t) - \frac{1}{2\eta}\|\widehat{y}_{t+1} - \widehat{y}_t\|^2 .$$

Combining and rearranging give

$$\begin{aligned}
\|\widehat{y}_t - \widehat{y}_{t+1}\|^2 &\le \eta(H(\widehat{y}_{t+1}) - H'(\widehat{y}_{t+1}) + H'(\widehat{y}_t) - H(\widehat{y}_t))\\
&= \eta(\langle \widehat{y}_t, z_t \rangle - \langle \widehat{y}_{t+1}, z_t \rangle)\\
&= \eta \langle \widehat{y}_t - \widehat{y}_{t+1}, z_t \rangle \le \eta \|\widehat{y}_t - \widehat{y}_{t+1}\| \, \|z_t\|_\star .
\end{aligned}$$

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

Combining this lemma with Equation (4) proves Theorem 2. Also note that the fact that z_t might depend on $\widehat{y}_t$ (which is indeed the case in the reduction of Equation (2)) does not affect the result, as mentioned earlier.

3 From Value to Algorithms — A General Recipe

By now we have seen quite a few online learning algorithms. One might wonder: how do we come up with an algorithm, especially when facing a new problem? Is there a general principle or even a concrete recipe to design algorithms?

To answer this question, we come back to the general setup and its minimax formulation:

$$\mathcal{V}^{\text{seq}}(\mathcal{F},n) = \left\langle\!\!\left\langle \inf_{q_t\in\Delta(\mathcal{D})} \sup_{z_t\in\mathcal{Z}} \mathbb{E}_{\widehat{y}_t\sim q_t} \right\rangle\!\!\right\rangle_{t=1}^{n} \left[\frac{\mathrm{Reg}(\mathcal{F},n)}{n}\right].$$

Previously, we derived a sequence of upper bounds on this minimax expression to study learnability, without having any concrete algorithms. However, in principle, one can be ambiguous and ask for the *exact minimax optimal* algorithm, which at time t predicts $\widehat{y}_t$ drawn from

$$\begin{aligned}
q_t &= \operatorname*{argmin}_{q_t\in\Delta(\mathcal{D})} \sup_{z_t\in\mathcal{Z}} \mathbb{E}_{\widehat{y}_t\sim q_t}\left[\left\langle\!\!\left\langle \inf_{q_\tau\in\Delta(\mathcal{D})} \sup_{z_\tau\in\mathcal{Z}} \mathbb{E}_{\widehat{y}_\tau\sim q_\tau} \right\rangle\!\!\right\rangle_{\tau=t+1}^{n} \mathrm{Reg}(\mathcal{F},n)\right] \\
&= \operatorname*{argmin}_{q_t\in\Delta(\mathcal{D})} \sup_{z_t\in\mathcal{Z}} \mathbb{E}_{\widehat{y}_t\sim q_t}\left[\left\langle\!\!\left\langle \inf_{q_\tau\in\Delta(\mathcal{D})} \sup_{z_\tau\in\mathcal{Z}} \mathbb{E}_{\widehat{y}_\tau\sim q_\tau} \right\rangle\!\!\right\rangle_{\tau=t+1}^{n} \left[\sum_{s=t}^{n}\ell(\widehat{y}_s,z_s) - \inf_{f\in\mathcal{F}}\sum_{s=1}^{n}\ell(f,z_s)\right]\right] \\
&= \operatorname*{argmin}_{q_t\in\Delta(\mathcal{D})} \sup_{z_t\in\mathcal{Z}} \left(\mathbb{E}_{\widehat{y}_t\sim q_t}\left[\ell(\widehat{y}_t,z_t)\right] + \left\langle\!\!\left\langle \inf_{q_\tau\in\Delta(\mathcal{D})} \sup_{z_\tau\in\mathcal{Z}} \mathbb{E}_{\widehat{y}_\tau\sim q_\tau} \right\rangle\!\!\right\rangle_{\tau=t+1}^{n} \left[\sum_{s=t+1}^{n}\ell(\widehat{y}_s,z_s) - \inf_{f\in\mathcal{F}}\sum_{s=1}^{n}\ell(f,z_s)\right]\right).
\end{aligned}$$

To simplify notation, recursively define the (unnormalized) *conditional value* of the game given the past decisions $z_{1:t}$ of the environment as

$$\begin{aligned}
&\mathcal{V}_n(z_{1:t}) = \inf_{q\in\Delta(\mathcal{D})} \sup_{z\in\mathcal{Z}} \left(\mathbb{E}_{\widehat{y}\sim q}\left[\ell(\widehat{y},z)\right] + \mathcal{V}_n(z_{1:t},z)\right) \\
\text{with}\qquad &\mathcal{V}_n(z_{1:n}) = -\inf_{f\in\mathcal{F}}\sum_{t=1}^{T}\ell(f,z_t).
\end{aligned}\tag{5}$$

In words, $\mathcal{V}_n(z_{1:t})$ is the optimal regret (offset by the loss already suffered) one can achieve against the worst-case future, given the past t steps. Clearly we have $\mathcal{V}^{\text{seq}}(\mathcal{F},n) = \frac{1}{n}\mathcal{V}_n(\emptyset)$. With this conditional value, the minimax strategy becomes

$$q_t = \operatorname*{argmin}_{q\in\Delta(\mathcal{D})} \sup_{z\in\mathcal{Z}} \left(\mathbb{E}_{\widehat{y}\sim q}\left[\ell(\widehat{y},z)\right] + \mathcal{V}_n(z_{1:t-1},z)\right).$$

In principle, finding q_t is a dynamic program. For most cases, however, there is no simple closed-form solution or efficient way to solve it exactly. Instead, we aim for finding an approximate solution. One general way to do so is to search for a *relaxation* of the conditional value. A relaxation Rel_n is a sequence of functions that map the past decisions of the environment $z_{1:t}$ to a real value for each $t = 1,\dots,n$ (just like the conditional value $\mathcal{V}_n$). It is called *admissible* if for any $z_1,\dots,z_T \in \mathcal{Z}$,

$$\begin{aligned}
\forall t = 1,\dots,n-1,\quad &\mathrm{Rel}_n(z_{1:t}) \ge \inf_{q\in\Delta(\mathcal{D})} \sup_{z\in\mathcal{Z}} \left(\mathbb{E}_{\widehat{y}\sim q}\left[\ell(\widehat{y},z)\right] + \mathrm{Rel}_n(z_{1:t},z)\right) \\
\text{and}\qquad &\mathrm{Rel}_n(z_{1:n}) \ge -\inf_{f\in\mathcal{F}}\sum_{t=1}^{T}\ell(f,z_t).
\end{aligned}\tag{6}$$

Comparing Equation (5) and Equation (6), it is clear that $\mathcal{V}_n(z_{1:t}) \le \mathrm{Rel}_n(z_{1:t})$ always holds, that is, admissible relaxation is indeed an upper bound of the conditional value. More importantly, if we replace the conditional value by the relaxation in the minimax optimal strategy, that is,

$$q_t = \operatorname*{argmin}_{q\in\Delta(\mathcal{D})} \sup_{z\in\mathcal{Z}} \left(\mathbb{E}_{\widehat{y}\sim q}\left[\ell(\widehat{y},z)\right] + \mathrm{Rel}_n(z_{1:t-1},z)\right),\tag{7}$$

then the regret of this algorithm is bounded by $\mathrm{Rel}_n(\emptyset)$. In fact, we do not even need to solve Equation (7) exactly — as long as q_t is such that

$$\sup_{z\in\mathcal{Z}} \left(\mathbb{E}_{\widehat{y}\sim q_t}\left[\ell(\widehat{y},z)\right] + \mathrm{Rel}_n(z_{1:t-1},z)\right) \le \mathrm{Rel}_n(z_{1:t}),\tag{8}$$

the regret is bounded by $\mathrm{Rel}_n(\emptyset)$ (note that the solution of Equation (7) always satisfies the above by admissibility). To see this, we just need to repeatedly peel off each term in the regret:

$$
\begin{aligned}
\mathbb{E}\left[\mathrm{Reg}(\mathcal{F}, n)\right] &\leq \mathbb{E}\left[\sum_{t=1}^{n} \ell(\widehat{y}_t, z_t) + \mathrm{Rel}_n(z_{1:n})\right] \\
&\leq \mathbb{E}\left[\sum_{t=1}^{n-1} \ell(\widehat{y}_t, z_t) + \sup_{z}\left(\mathbb{E}_{\widehat{y}\sim q_n}\left[\ell(\widehat{y}, z)\right] + \mathrm{Rel}_n(z_{1:n-1}, z)\right)\right] \\
&\leq \mathbb{E}\left[\sum_{t=1}^{n-1} \ell(\widehat{y}_t, z_t) + \mathrm{Rel}_n(z_{1:n-1})\right] \leq \cdots \leq \mathrm{Rel}_n(\emptyset).
\end{aligned}
$$

Therefore, as long as we can find an admissible relaxation that is easy to compute and that is not too large, we have a reasonable algorithm. But how to we find such a good relaxation? In fact, in some sense we have seen one already while discussing learnability. Recall that the value of the game is bounded by (twice of) the sequential Rademacher complexity, which after scaling means $\mathcal{V}_n(\emptyset) \leq \sup_{\boldsymbol{z}} \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{f\in\mathcal{F}} 2\sum_{t=1}^{n} \epsilon_t \ell(f, \boldsymbol{z}_t(\boldsymbol{\epsilon}))\right]$. We generalize the concept of sequential Rademacher complexity and define:

$$
\mathcal{R}_n(z_{1:t}) = \sup_{\boldsymbol{z}} \mathbb{E}_{\epsilon_{t+1:n}} \sup_{f\in\mathcal{F}} \left(2\sum_{s=t+1}^{n} \epsilon_s \ell(f, \boldsymbol{z}_{s-t}(\epsilon_{t+1:s-1})) - \sum_{s=1}^{t} \ell(f, z_s)\right), \tag{9}
$$

where $\boldsymbol{z}$ ranges over all $\mathcal{Z}$-valued tree with depth $n-t$, which represents the worst-case future. It can be shown that sequential Rademacher complexity is indeed an admissible relaxation.

Theorem 3. *The generalized sequential Rademacher complexity Equation (9) is an admissible relaxation.*

The proof is based on the same symmetrization technique we have discussed before. Roughly speaking, the conditional value given $z_{1:t}$ is in terms of the difference between the total future loss of the learner and the benchmark. The first t terms of the benchmark appear as the second summation in Equation (9), while the last $n-t$ terms is combined with the future total loss of the learner via symmetrization to become the first summation of Equation (9). We leave the complete proof as an exercise.

With this relaxation, we can assert that the strategy defined in Equation (8) is definitely a good algorithm in terms of having low regret — as mentioned the regret is bounded as $\mathrm{Rel}_n(\emptyset) = \mathcal{R}_n(\emptyset)$, which is exactly the original sequential Rademacher complexity (scaled by n) and is very close to the value of the game according to previous lectures. However, is there a way to efficiently compute this relaxation? Unfortunately the answer is usually no, especially due to the part $\sup_{\boldsymbol{z}}$. Nevertheless, this relaxation is already much manageable compared to the conditional value, and very often, further bounding this relaxation via simple algebra will lead to another relaxation that is efficiently computable and at the same time small enough. This gives a general "recipe" to design an online learning algorithm:

1. Start with the sequential Rademacher complexity Equation (9).
2. Derive an upper bound of it to get a relaxation that is easy to compute.
3. Prove that the relaxation is admissible.
4. Derive the final algorithm using Equation (7) or Equation (8).

Figure 2: A general recipe to derive an online learning algorithm

Let's see a concrete example. Consider the case when the loss function is linear: $\ell(f, z) = \langle f, z\rangle$, which is representative for convex losses per earlier discussions. Further restrict our attention to $\mathcal{F} = \mathcal{Z} = B_2^d$, the unit ℓ_2 ball. Then the generalized sequential Rademacher complexity reduces to

$$
\mathcal{R}_n(z_{1:t}) = \sup_{\boldsymbol{z}} \mathbb{E}_{\epsilon_{t+1:n}} \sup_{f\in B_2^d} \left\langle f, 2\sum_{s=t+1}^{n} \epsilon_s \boldsymbol{z}_{s-t}(\epsilon_{t+1:s-1}) - \sum_{s=1}^{t} z_s \right\rangle
$$

$$= \sup_{\boldsymbol{z}} \mathbb{E}_{\epsilon_{t+1:n}} \left\| 2 \sum_{s=t+1}^{n} \epsilon_s \boldsymbol{z}_{s-t}(\epsilon_{t+1:s-1}) - \sum_{s=1}^{t} z_s \right\|_2 .$$

Using Jensen's equality, we upper bound it as

$$\begin{aligned}
\mathcal{R}_n(z_{1:t}) &\le \sup_{\boldsymbol{z}} \sqrt{\mathbb{E}_{\epsilon_{t+1:n}} \left\| 2 \sum_{s=t+1}^{n} \epsilon_s \boldsymbol{z}_{s-t}(\epsilon_{t+1:s-1}) - \sum_{s=1}^{t} z_s \right\|_2^2} \\
&= \sup_{\boldsymbol{z}} \sqrt{4 \sum_{s=t+1}^{n} \|\boldsymbol{z}_{s-t}(\epsilon_{t+1:s-1})\|_2^2 + \left\| \sum_{s=1}^{t} z_s \right\|_2^2} \qquad \text{(all other terms have zero mean)} \\
&\le \sqrt{4(n-t) + \left\| \sum_{s=1}^{t} z_s \right\|_2^2},
\end{aligned}$$

and take the last (very simple) expression as the relaxation $\mathrm{Rel}_n(z_{1:t})$. Next we prove that this is indeed admissible. The second line of Equation (6) holds with equality. To show the first line, it is enough to consider a proper strategy such that

$$\begin{aligned}
&\inf_{q \in \Delta(\mathcal{D})} \sup_{z \in \mathcal{Z}} \left(\mathbb{E}_{\widehat{y} \sim q} \left[\ell(\widehat{y}, z) \right] + \mathrm{Rel}_n(z_{1:t}, z) \right) = \inf_{\widehat{y} \in B_2^d} \sup_{z \in B_2^d} \left(\langle \widehat{y}, z \rangle + \sqrt{4(n-t-1) + \left\| \sum_{s=1}^{t} z_s + z \right\|_2^2} \right) \\
&\le \inf_{\widehat{y} \in B_2^d} \sup_{z \in B_2^d} \left(\langle \widehat{y}, z \rangle + \sqrt{4(n-t) + \left\| \sum_{s=1}^{t} z_s \right\|_2^2 + 2 \left\langle \sum_{s=1}^{t} z_s, z \right\rangle} \right) .
\end{aligned}$$

Note that the optimal $\widehat{y}$ has to be in the same direction as $\sum_{s=1}^{t} z_s$, for otherwise, $\widehat{y}$ has some component that is perpendicular to $\sum_{s=1}^{t} z_s$, and the environment can also add a component in the same direction to increase the term $\langle \widehat{y}, z \rangle$, while keeping the rest unchanged. Therefore, we let $\widehat{y} = -\alpha \sum_{s=1}^{t} z_s$ for some coefficient $\alpha \in \mathcal{A} = (0, 1/\|\sum_{s=1}^{t} z_s\|_2]$ (such that $\widehat{y} \in B_2^d$) and arrive at

$$\begin{aligned}
&\inf_{q \in \Delta(\mathcal{D})} \sup_{z \in \mathcal{Z}} \left(\mathbb{E}_{\widehat{y} \sim q} \left[\ell(\widehat{y}, z) \right] + \mathrm{Rel}_n(z_{1:t}, z) \right) \\
&\le \inf_{\alpha \in \mathcal{A}} \sup_{z \in B_2^d} \left(-\alpha \left\langle \sum_{s=1}^{t} z_s, z \right\rangle + \sqrt{4(n-t) + \left\| \sum_{s=1}^{t} z_s \right\|_2^2 + 2 \left\langle \sum_{s=1}^{t} z_s, z \right\rangle} \right) \\
&\le \inf_{\alpha \in \mathcal{A}} \sup_{\beta \in \mathbb{R}} \left(-\alpha\beta + \sqrt{4(n-t) + \left\| \sum_{s=1}^{t} z_s \right\|_2^2 + 2\beta} \right) \\
&= \inf_{\alpha \in \mathcal{A}} \frac{1}{2} \left(\frac{1}{\alpha} + \alpha \left(4(n-t) + \left\| \sum_{s=1}^{t} z_s \right\|_2^2 \right) \right) = \sqrt{4(n-t) + \left\| \sum_{s=1}^{t} z_s \right\|_2^2} = \mathcal{R}_n(z_{1:t}).
\end{aligned}$$

This shows that the relaxation is indeed admissible, and in fact also gives an algorithm

$$\widehat{y}_{t+1} = -\alpha_t \sum_{s=1}^{t} z_s, \quad \text{where} \quad \alpha_t = \frac{1}{\sqrt{4(n-t) + \left\| \sum_{s=1}^{t} z_s \right\|_2^2}},$$

whose regret is bounded by $\mathrm{Rel}_n(\emptyset) = 2\sqrt{n}$. Note that this is very close to the gradient descent algorithm discussed in Section 2, but without any explicit projection or learning rate.

This is just one simple example to illustrate the power of the general recipe. Using this framework one can in fact recover many other algorithms, such as FTRL (and hence Hedge as well), and also derive others that were not known before. This shows that existing algorithms are not "methods that just work" — they can in fact all be derived in a principled way, starting from the fundamental sequential Rademacher complexity.