Theoretical Machine Learning
Lecture 1

Instructor: Haipeng Luo

1 A Gentle Start: Supervised Learning

One of the most common learning problems is supervised learning. Below we will use this as an example to see how to formally and meaningfully define a learning problem.

In a supervised learning problem, we are given a training set consisting of input-output pairs (often called *examples*), and our goal is to learn from these examples some pattern on the connection between input and output, and to come up with a good *predictor* that hopefully could accurately predict the output of an unseen input. Examples include image classification (input = picture, output = dog or cat), machine translation (input = English, output = French), video summarization (input = video, output = caption), and many more.

What is the principle of designing such learning procedures, and how do we know if such learning will succeed or not? To answer these questions, we need to first formalize the learning problem. Let $\mathcal{X}$ and $\mathcal{Y}$ be some arbitrary input and output space, and $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ be a training set of n examples. For instance, in the image classification example, $\mathcal{X}$ might the space of all 256×256 images, and $\mathcal{Y}$ consists of two outcomes: dog and cat. It is often more convenient to further abstract these spaces. For example, we could represent an image by a vector in $\mathbb{R}^d$ for some dimension d, and use -1 to represent label "dog" and $+1$ to represent label "cat", so that $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$.

Our goal is to come up with a predictor $\widehat{y} \in \mathcal{Y}^{\mathcal{X}}$, which is a function mapping from the input space to the output space. Given a new unseen input $x \in \mathcal{X}$, the predictor predicts $\widehat{y}(x)$ as the output. For example, $\widehat{y}$ could be a linear classifier, a decision tree, or a neural net.

So how do we measure the accuracy of $\widehat{y}$? Suppose the actual output corresponding to x is y, then the accuracy should naturally be measured by comparing y and $\widehat{y}(x)$ in some way. To this end, define a general loss function $\ell : \mathcal{Y}^{\mathcal{X}} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$, which maps a predictor and an input-output pair to some loss value. The larger this value, the less accurate the predictor is on this example.

Some loss functions might be more suitable than others for a specific task. For example, for binary classification (e.g. predict dog or cat for an image) with $\mathcal{Y} = \{-1, +1\}$, a very natural loss is $\ell(\widehat{y}, (x, y)) = \mathbb{I}\{\widehat{y}(x) \neq y\}$, which is 1 if the predictor predicts a different label, and 0 otherwise (so-called 0-1 loss). On the other hand, for regression (e.g. predicting house price), usually we have $\mathcal{Y} = \mathbb{R}$ and the loss is defined as $\ell(\widehat{y}, (x, y)) = (\widehat{y}(x) - y)^2$, which is the squared difference between the prediction and the true output (so-called square loss).

Apparently measuring the accuracy/loss of the predictor on one single example does not sound like a good idea. In practice, we often measure the average loss over a *test set*: $(x'_1, y'_1), \ldots, (x'_m, y'_m) \in \mathcal{X} \times \mathcal{Y}$ for some m, defined as $\frac{1}{m} \sum_{i=1}^{m} \ell(\widehat{y}, (x'_i, y'_i))$ and often called the test error. However, so far the problem is clearly still not well-defined — without any connection between the training set and the test set, how is it possible at all to learn a predictor with small test error?

i.i.d. assumption. Note that what usually happens in practice is that we collect a bunch of examples from the nature/environment, and then randomly split them into a training set and a test set. This is often modeled as an i.i.d. setting, where it is assumed that examples from the nature are

generated independently according to a fixed but *unknown* distribution $\mathcal{P}$ supported on $\mathcal{X} \times \mathcal{Y}$. In this case, the training set and the test set are clearly related — the examples in these set are all i.i.d. samples of $\mathcal{P}$.

With this assumption, it makes more sense to measure the quality of a predictor $\widehat{y}$ by its *expected* test error: $\mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\ell(\widehat{y},(x,y))\right]$, where the expectation is with respect to the random draw of a new example from the distribution $\mathcal{P}$. This is often called the risk or the generalization error. For a fixed predictor $\widehat{y}$, average loss over a test set is clearly just an unbiased estimate of the risk and it is easy to compute in practice. However, in theory it is more suitable to use the risk as the measure since this removes the extra randomness from the test set. Indeed, for a *fixed* predictor $\widehat{y}$, $\mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\ell(\widehat{y},(x,y))\right]$ is a fixed quantity while the test error $\frac{1}{m}\sum_{i=1}^{m}\ell(\widehat{y},(x_i',y_i'))$ is a random variable. We also emphasize that $\mathcal{P}$ is unknown in this formulation (which is of course also the case in practice), otherwise finding a predictor with small risk would simply be an optimization problem instead of a learning problem.

So with this i.i.d. assumption, is the learning problem well-defined now? Not completely — the problem is still too general to be meaningful. Too see this, simply consider a binary classification problem, where $\mathcal{P}(y=+1\mid x)$ is $\frac{1}{2}$ for every x, that is, the label of every input is a fair coin flip according to the distribution $\mathcal{P}$ that generates the examples. In this case, *no matter what $\widehat{y}$ is*, the expected 0-1 loss $\mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\mathbb{I}\{\widehat{y}(x)\neq y\}\right]$ is simply $\frac{1}{2}$, so no learning will ever be possible.

Classical Statistic vs Statistical Learning. Of course, the above example is very pathological and does not reflect what really happens in practice. To make the problem meaningful, we thus need to incorporate more prior knowledge on the problem, and here comes the key separation between classical statistic and statistical learning. In classical statistic, the standard approach is to assume some very specific structure on the data distribution $\mathcal{P}$. Take regression for example, one might make the assumption that the marginal distribution $\mathcal{P}_{\mathcal{X}}$ over $\mathcal{X}$ is a Gaussian distribution with unknown mean and covariance, and the conditional distribution $\mathcal{P}(y\mid x)$ is also a Gaussian with mean $\langle\theta,x\rangle$ and some unknown variance, for some unknown parameter θ. Under such a structural assumption, a natural approach would be to estimate all the unknown parameters using the training set (via Maximum Likelihood Estimation for example), and with the estimated parameters, prediction becomes pretty easy.

It is not hard to imagine that such a approach would work well if the assumption on $\mathcal{P}$ indeed holds. However, it often provides no guarantees if the assumption is far away from the reality. To avoid making such a strong assumption, statistical learning takes a quite different approach, which is often called *agnostic* or *distribution-free*. It shifts the focus from the data-generating distribution to some reference class of models $\mathcal{F}\subset\mathcal{Y}^{\mathcal{X}}$, and asks the question: can I learn a predictor that is reasonably well compared to the best option from the reference class $\mathcal{F}$, without making any assumption on $\mathcal{P}$? In other words, we would like to make sure that the difference between the risk of $\widehat{y}$ and that of the best fixed predictor from $\mathcal{F}$, $\mathbb{E}\left[\ell(\widehat{y},(x,y))\right]-\inf_{f\in\mathcal{F}}\mathbb{E}\left[\ell(f,(x,y))\right]$, is relatively small, and we want this to be true for any distribution $\mathcal{P}$ (hence distribution-free).

The rational behind this goal is that if we believe that the reference class $\mathcal{F}$ is good enough to ensure a small risk, then our predictor is also reasonably good. So instead of incorporating the prior knowledge into the structure of $\mathcal{P}$, we incorporate the prior knowledge into the process of selecting $\mathcal{F}$. However, importantly this is in some sense a more "robust" approach, as we never impose an explicit assumption on $\mathcal{F}$ or $\mathcal{P}$, and in particular we do not require the ground truth (formally the Bayes predictor) to be in $\mathcal{F}$.

This agnostic formulation of learning problem will be the key focus of this course, and we will make it even more formal in the following sections. Of course one might still ask if the problem is well-defined now and if such an agnostic approach exists. The answer will (naturally) depend on the expressiveness of the reference class $\mathcal{F}$, which also determines the complexity of learning, and this is one of the main subjects of this course.

2 A General Setup: Statistical Learning

Having the above supervised learning example in mind, now we introduce a slightly more general statistical learning setup that can capture more problems beyond supervised learning. Instead of using an input-output pair (x,y) to represent an example, we will use a more abstract notation $z\in\mathcal{Z}$

instead, for some abstract space $\mathcal{Z}$. A training set of size n is generated by drawing n independent samples $z_1, \ldots, z_n \in \mathcal{Z}$ from a fixed distribution $\mathcal{P}$ that is unknown to the learner. After seeing the training set, the learner needs to come up with a predictor $\widehat{y} \in \mathcal{D}$ for some arbitrary decision space $\mathcal{D}$. It is worth pointing out that the notation $\widehat{y}$ in fact hides the dependence on the training set $z_{1:n}$.

The loss function ℓ is now a mapping from $\mathcal{D} \times \mathcal{Z}$ to $\mathbb{R}$, and the risk of a predictor $\widehat{y}$ is defined as $L(\widehat{y}) = \mathbb{E}_{z\sim\mathcal{P}}[\ell(\widehat{y}, z)]$. It is important to note that $L(\widehat{y})$ is a *random variable* when $\widehat{y}$ is the output of the learner, simply because $\widehat{y}$ depends on the training set, which itself is randomly generated. The expected risk of $\widehat{y}$ should thus be written as $\mathbb{E}_{z_{1:n}\sim\mathcal{Z}^n}[L(\widehat{y})] = \mathbb{E}_{z_{1:n}\sim\mathcal{Z}^n}[\mathbb{E}_{z\sim\mathcal{P}}[\ell(\widehat{y}, z)]]$, but whenever there is no confusion, we will simply use the notation $\mathbb{E}[L(\widehat{y})]$ or even $\mathbb{E}[\ell(\widehat{y}, z)]$, where the expectation is with respect to the randomness of the training set, the unseen test point z, and in fact even the internal randomness of the learner.

According to previous discussion, we will compare the expected risk of $\widehat{y}$ to the smallest risk achieved by some reference space $\mathcal{F} \subset \mathcal{D}$, which is called the *excess risk* and is defined as: $\mathbb{E}[L(\widehat{y})] - \inf_{f\in\mathcal{F}} L(f)$. When $\mathcal{F} = \mathcal{D}$, the learner is called *proper*; otherwise, the learner is *improper*. We will mostly consider proper learners in this course. The learner's goal is to come up with a strategy that ensures small excess risk, and here "small" means that the excess risk goes down to 0 when n goes to infinity. In other words, the learner needs to find a predictor whose risk is arbitrarily close to that of the best predictor in $\mathcal{F}$, as the number of training examples increases. If such an algorithm exists, we say that $\mathcal{F}$ is *learnable*.

2.1 Examples

Many common learning problems can be captured by the setting described above. We already discussed agnostic supervised learning, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\mathcal{D} \subset \mathcal{Y}^{\mathcal{X}}$ consists of mappings from input to output, such as decision trees or neural nets. For classification, $\mathcal{Y}$ is a discrete set and 0-1 loss is commonly used, while for regression, $\mathcal{Y}$ is usually a continuous subset of $\mathbb{R}$ and the square loss is commonly used.

PAC setting. *Probably Approximately Correct* (PAC) is a fundamental learning framework that can be considered as the start of the field of computational learning theory. The most basic PAC setting considers a supervised binary classification problem with $\mathcal{Y} = \{-1, +1\}$, and makes the assumption that $\mathcal{P}(y = f^\star(x) \mid x) = 1$ for some $f^\star \in \mathcal{F}$. In other words, the label is realized deterministically by a fixed ground truth function in the reference class. Note that in this case for 0-1 loss we have $\inf_{f\in\mathcal{F}} L(f) = 0$ and the excess risk is simply the risk of the learner's output $\widehat{y}$. Instead of considering the expected risk of $\widehat{y}$, in PAC we ask if one can come up with an algorithm such that for any given $\epsilon > 0$, any confidence level $\delta > 0$, any marginal distributions $\mathcal{P}_\mathcal{X}$, and any $f^\star \in \mathcal{F}$, after seeing $\mathrm{poly}(\frac{1}{\epsilon}, \frac{1}{\delta})$ training examples, the output $\widehat{y}$ satisfies $\mathcal{P}(L(\widehat{y}) \le \epsilon) \ge 1 - \delta$. If such a algorithm exists, $\mathcal{F}$ is called PAC-learnable. It turns out that whether a class is PAC-learnable is determined by similar things that determines the learnability of a general statistical learning problem, and we will thus focus on the more general setup.

Density estimation. So far all examples are instances of supervised learning. Here we consider an unsupervised learning example, where the goal of the learner is to estimate the density of the data-generating distribution $\mathcal{P}$. In particular, $\mathcal{D} = \mathcal{F}$ consists of density functions supported on $\mathcal{Z}$, and the common loss function is the *log loss* $\ell(\widehat{y}, z) = -\log \widehat{y}(z)$. The rational behind log loss is that now the excess risk is connected to the Kullback-Leibler (KL) divergence (with a slight abuse of notation, we use $\mathcal{P}$ to denote its density as well):

$$\begin{aligned} L(\widehat{y}) - L(f) &= -\mathbb{E}_{z\sim\mathcal{P}}[\log \widehat{y}(z)] + \mathbb{E}_{z\sim\mathcal{P}}[\log f(z)] \\ &= \mathbb{E}_{z\sim\mathcal{P}}[\log \mathcal{P}(z)] - \mathbb{E}_{z\sim\mathcal{P}}[\log \widehat{y}(z)] + \mathbb{E}_{z\sim\mathcal{P}}[\log f(z)] - \mathbb{E}_{z\sim\mathcal{P}}[\log \mathcal{P}(z)] \\ &= \mathrm{KL}(\mathcal{P} \,\|\, \widehat{y}) - \mathrm{KL}(\mathcal{P} \,\|\, f). \end{aligned}$$

2.2 The value of the game and no free lunch

By now you probably have a feeling that a learning problem can really be treated as a (zero-sum) game between a learner and an environment. The learner decides a learning strategy first, and then the environment decides a data-generating distribution and generates a training set. How well the

learner is doing is measured by the excess risk. More precisely, we turn our focus to the following minimax quantity, called the value of this game:

$$\mathcal{V}^{\text{iid}}(\mathcal{F}, n) = \inf_{\pi} \sup_{\mathcal{P}} \left(\mathbb{E}\left[L(\widehat{y})\right] - \inf_{f \in \mathcal{F}} L(f) \right).$$

Here, $\mathcal{P}$ ranges over all distributions over $\mathcal{Z}$, and π ranges over all strategies of the learner, that is, all mappings from n training examples to a predictor $\widehat{y} \in \mathcal{D}$ if the learner has no internal randomness, or all distributions over mappings from n training examples to a predictor $\widehat{y} \in \mathcal{D}$ if the learner is randomized. Note that it is important that the learner "acts" first (that is, $\inf \sup$ instead of $\sup \inf$), which corresponds to the fact that the learner's strategy needs to work for *all* distributions.

Clearly, the statement that $\mathcal{F}$ is learnable is equivalent to saying $\limsup_{n\to\infty} \mathcal{V}^{\text{iid}}(\mathcal{F}, n) = 0$. In other words, studying the value $\mathcal{V}^{\text{iid}}(\mathcal{F}, n)$ is all we need to do to understand the learnability of a class $\mathcal{F}$. This, however, does not directly give us an algorithm for learning $\mathcal{F}$ though (when it is learnable).

Is every class learnable? The answer is no, probably unsurprisingly. The following so-called no free lunch theorem shows that one cannot learn a class that is too general.

Theorem 1 (No Free Lunch)**.** *Suppose $|\mathcal{X}| \geq 2n$, $\mathcal{Y} = \{-1, +1\}$, and $\ell(\widehat{y}, (x, y)) = \mathbb{I}\{\widehat{y}(x) \neq y\}$ is the 0-1 loss. We have $\mathcal{V}^{\text{iid}}(\mathcal{Y}^{\mathcal{X}}, n) \geq 1/4$.*

Proof. We will restrict the environment to the following simple family of distributions $\mathcal{P}^f$: the marginal distribution $\mathcal{P}^f_{\mathcal{X}}$ is uniform over a fixed subset $\mathcal{X}' \subset \mathcal{X}$ with $|\mathcal{X}'| = 2n$, and the conditional distribution is parameterized by some $f \in \mathcal{F}' = \left\{ f \in \mathcal{Y}^{\mathcal{X}} : f(x) = -1,\ \forall x \notin \mathcal{X}' \right\}$ so that $\mathcal{P}^f(y = f(x) \mid x) = 1$. We thus have

$$\begin{aligned}
\mathcal{V}^{\text{iid}}(\mathcal{F}, n) &= \inf_{\pi} \sup_{\mathcal{P}} \left(\mathbb{E}\left[L(\widehat{y})\right] - \inf_{f \in \mathcal{F}} L(f) \right) \\
&\geq \inf_{\pi} \sup_{\mathcal{P}^f : f \in \mathcal{F}'} \left(\mathbb{E}\left[L(\widehat{y})\right] - \inf_{f \in \mathcal{F}} L(f) \right) \\
&= \inf_{\pi} \sup_{\mathcal{P}^f : f \in \mathcal{F}'} \mathbb{E}\left[L(\widehat{y})\right] && (\inf_{f\in\mathcal{F}} L(f) = 0 \text{ for any } \mathcal{P}^f) \\
&\geq \inf_{\pi} \mathbb{E}_{f\sim\mathcal{Q}} \mathbb{E}\left[L(\widehat{y})\right]. && (\mathcal{Q} \text{ is uniform over } \mathcal{F}')
\end{aligned}$$

The last step holds because clearly sup is not less than any weighted average. This step is in fact the key of the proof — moving from sup to average will eventually allow us to ignore the behavior of the learner, a powerful technique in proving lower bounds. Indeed, to make the dependence clear, let S_n be the training set and write $\widehat{y}(\cdot; S_n)$ as the output of the learner after seeing the training set. We continue with

$$\begin{aligned}
\mathbb{E}_{f\sim\mathcal{Q}} \mathbb{E}\left[L(\widehat{y})\right] &= \mathbb{E}_{f\sim\mathcal{Q}} \mathbb{E}_{x, x_{1:n}} \left[\mathbb{I}\{\widehat{y}(x; S_n) \neq f(x)\}\right] \\
&= \mathbb{E}_{x, x_{1:n}} \mathbb{E}_{f\sim\mathcal{Q}} \left[\mathbb{I}\{\widehat{y}(x; S_n) \neq f(x)\}\right] \\
&\geq \mathcal{P}_{\mathcal{X}}\left(x \notin x_{1:n}\right) \mathbb{E}_{x, x_{1:n}} \mathbb{E}_{f\sim\mathcal{Q}} \left[\mathbb{I}\{\widehat{y}(x; S_n) \neq f(x)\} \mid x \notin x_{1:n}\right] \\
&\geq \frac{1}{2} \mathbb{E}_{x, x_{1:n}} \mathbb{E}_{f\sim\mathcal{Q}} \left[\mathbb{I}\{\widehat{y}(x; S_n) \neq f(x)\} \mid x \notin x_{1:n}\right],
\end{aligned}$$

where the last step is because $x_{1:n}$ is a subset of $\mathcal{X}'$ with at most n distinct elements and x is uniformly at random drawn from $\mathcal{X}'$. The proof is completed by realizing that $\mathbb{E}_{f\sim\mathcal{Q}} \left[\mathbb{I}\{\widehat{y}(x; S_n) \neq f(x)\} \mid x \notin x_{1:n}\right] \geq 1/2$, because conditioning on the label of the training set (and the randomness of the learner), $\widehat{y}(x; S_n)$ is fixed while $f(x)$ is $+1$ (or -1) with probability $1/2$, by the definition of $\mathcal{Q}$. □

Again, this theorem shows that there is no hope to learn a class that is too expressive. In the next lecture we will study how to measure the "expressiveness" of a class.

3 A Harder Setup: Online Learning

The i.i.d. assumption of statistical learning, while standard, might be too strong in some cases. There are many works on relaxing this condition for statistical learning (for example, assume the

training set and test set are from related but different distributions), but this is out of the scope of this course. Instead, we will focus on a rather difficult setting called online learning (or sequential prediction, sequential decision making, online optimization, etc.), which completely removes any distributional assumptions.

We will follow most notation from the statistical learning setup. The key difference is in the learning protocol. For statistical learning, a batch of data is available ahead of time, and learning is essentially making a one-shot decision (that is, coming up with $\widehat{y}$). Statistical learning is also call batch learning sometimes because of this fact. For online learning, however, data are presented one by one in a sequential manner, and the learner is asked to make a sequence of decisions. More concretely, the game proceeds in rounds, and for each round $t = 1, \dots, n$,

- the learner predicts $\widehat{y}_t \in \mathcal{D}$ while the environment chooses $z_t \in \mathcal{Z}$ simultaneously,
- the learner suffers loss $\ell(\widehat{y}_t, z_t)$ and observes z_t.

All the examples we discussed earlier for statistical learning have an online analogue. Such online formulation indeed captures some real-world applications better, especially for Internet applications that are everywhere nowadays. For example, email spam detection, recommendation systems, search, etc. are all arguably better captured by an online formulation.

Very similar to the definition of excess risk, in online learning we measure the performance of the learning by the *regret*, which is the difference between the learner's total loss and that of the best fixed predictor from a reference class $\mathcal{F}$ in hindsight:

$$\mathrm{Reg}(\mathcal{F}, n) = \sum_{t=1}^{n} \ell(\widehat{y}_t, z_t) - \inf_{f \in \mathcal{F}} \sum_{t=1}^{n} \ell(f, z_t).$$

So average regret is similar to excess risk, and we would like design an online learning algorithm that ensures the average regret goes down to 0 as n increases. However, a very important distinction is that there is usually no distributional assumption on how $z_1, \dots, z_n$ are generated. In fact, they could even be chosen by a malicious adversary! But what does that really mean mathematically? What can z_t depend on?

To answer this question, we first make it clear what $\widehat{y}_t$ can depend on. Naturally, $\widehat{y}_t$ can depend on $z_1, \dots, z_{t-1}$, the previous outcomes from the environment before round t. In addition, if the learner is randomized (which in fact is necessary in some cases), $\widehat{y}_t$ will depend on the internal randomness of the learner as well. Equivalently, we can define the learner's strategy π as a distribution over a sequence of mappings $\pi_t : \mathcal{Z}^{t-1} \to \mathcal{D}$ for $t = 1, \dots, T$.

Now, depending on what z_t can depend on, we can define two kinds of environments. The first one is called oblivious environment, where $z_1, \dots, z_T$ can only depend on π, but not directly on the learner's decisions $\widehat{y}_1, \dots, \widehat{y}_T$. In other words, we can imagine the environment in fact decides the entire sequence of outcomes $z_1, \dots, z_T$ even before the game starts, knowing the algorithm of the learner. The second one is called adaptive environment, where $z_1, \dots, z_T$ can again depend on π, and in addition z_t can depend on $\widehat{y}_1, \dots, \widehat{y}_{t-1}$, the previous decisions of the learner before round t.

Clearly, adaptive environments are harder than oblivious environments, from the learner's viewpoint. Adaptive environments nicely capture applications where the opponent might be malicious, such as spam detection. Somewhat surprisingly though, in most cases the difference in learnability between adaptive and oblivious environments is not that crucial.

3.1 The value of the game and online-to-batch conversion

Now that the problem is well defined, we should ask the question again: is learning possible at all for such a difficult problem? Similarly to statistical learning, we turn our focus to the value of the game:

$$\mathcal{V}^{\mathrm{seq}}(\mathcal{F}, n) = \inf_{\pi} \sup_{z_{1:n}} \mathbb{E}\left[\frac{\mathrm{Reg}(\mathcal{F}, n)}{n}\right] = \inf_{\pi} \sup_{z_{1:n}} \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \ell(\widehat{y}_t, z_t) - \inf_{f \in \mathcal{F}} \frac{1}{n}\sum_{t=1}^{n} \ell(f, z_t)\right],$$

where it is understood that for oblivious environments, $z_{1:n}$ ranges over $\mathcal{Z}^n$, while for adaptive environments, each z_t ranges over all mappings from $\mathcal{D}^{t-1}$ to $\mathcal{Z}$. We call $\mathcal{F}$ *online learnable* if

$\limsup_{n\to\infty} \mathcal{V}^{\text{seq}}(\mathcal{F}, n) = 0$. An algorithm that ensures vanishing regret as n increases is sometimes called a *no-regret* algorithm.

We argue that for adaptive environments, the value of the game can in fact be written as a sequence of minimax expressions:

$$\mathcal{V}^{\text{seq}}(\mathcal{F}, n) = \inf_{q_1 \in \Delta(\mathcal{D})} \sup_{z_1 \in \mathcal{Z}} \mathbb{E}_{\widehat{y}_1 \sim q_1} \cdots \inf_{q_n \in \Delta(\mathcal{D})} \sup_{z_n \in \mathcal{Z}} \mathbb{E}_{\widehat{y}_n \sim q_n} \left[\frac{\text{Reg}(\mathcal{F}, n)}{n} \right] \tag{1}$$

where $\Delta(\mathcal{D})$ is the simplex over $\mathcal{D}$. We omit the (somewhat tedious) proof for this fact, but you should be able to convince yourself that this is true. This alternative expression of the value will be essential for further developments in the future lectures. For notational convenience, we deploy the following shorthand to suppress the long minimax sequence:

$$\mathcal{V}^{\text{seq}}(\mathcal{F}, n) = \left\langle\!\!\left\langle \inf_{q_t \in \Delta(\mathcal{D})} \sup_{z_t \in \mathcal{Z}} \mathbb{E}_{\widehat{y}_t \sim q_t} \right\rangle\!\!\right\rangle_{t=1}^{n} \left[\frac{\text{Reg}(\mathcal{F}, n)}{n} \right]. \tag{2}$$

We conclude this section by proving an intuitive statement: online learning is harder than statistical learning, that is $\mathcal{V}^{\text{iid}}(\mathcal{F}, n) \le \mathcal{V}^{\text{seq}}(\mathcal{F}, n)$.

Theorem 2. *For any $\mathcal{F}$ and any n, we have $\mathcal{V}^{\text{iid}}(\mathcal{F}, n) \le \mathcal{V}^{\text{seq}}(\mathcal{F}, n)$.*

Proof. The statement is proven via a classical online-to-batch conversion, which states that given any online strategy, one can convert it into a batch strategy with excess risk at most the average regret of the online strategy. This clearly implies the statement.

The conversion works as follows. Given a training set $z_1, \ldots, z_n$ in the statistical learning setting, and an algorithm π for the online setting, simply feed $z_1, \ldots, z_n$ one by one to π and obtain decisions $\widehat{y}_1, \ldots, \widehat{y}_n$. Finally, uniformly at random pick one of these decisions as the final predictor $\widehat{y}$.

For any $\epsilon > 0$, let f_ϵ be such that $L(f_\epsilon) \le \inf_{f \in \mathcal{F}} L(f) + \epsilon$, which must exist by the definition of $\inf$. The excess risk of $\widehat{y}$ is then bounded as

$$\begin{aligned}
\mathbb{E}\left[L(\widehat{y})\right] - \inf_{f \in \mathcal{F}} L(f) &\le \mathbb{E}\left[L(\widehat{y})\right] - L(f_\epsilon) + \epsilon && \text{(by definition of } f_\epsilon\text{)} \\
&= \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}\left[L(\widehat{y}_t)\right] - \frac{1}{n}\sum_{t=1}^{n} L(f_\epsilon) + \epsilon && \text{(by the construction of } \widehat{y}\text{)} \\
&= \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \ell(\widehat{y}_t, z_t) - \frac{1}{n}\sum_{t=1}^{n} \ell(f_\epsilon, z_t)\right] + \epsilon \\
&\le \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \ell(\widehat{y}_t, z_t) - \inf_{f \in \mathcal{F}} \frac{1}{n}\sum_{t=1}^{n} \ell(f, z_t)\right] + \epsilon
\end{aligned}$$

where the last equality uses the fact that $\widehat{y}_t$ and f_ϵ do not depend on z_t. Finally, since the above holds for *all* ϵ, one must have

$$\mathbb{E}\left[L(\widehat{y})\right] - \inf_{f \in \mathcal{F}} L(f) \le \mathbb{E}\left[\frac{1}{n}\sum_{t=1}^{n} \ell(\widehat{y}_t, z_t) - \inf_{f \in \mathcal{F}} \frac{1}{n}\sum_{t=1}^{n} \ell(f, z_t)\right],$$

that is, the excess risk is never larger than the expected average regret, finishing the proof. □

In a few lectures, we will see that this inequality is in fact strict, that is, there exist classes that are learnable in the statistical learning setting but not learnable in the online setting.

4 An Even Harder Setup: Online Learning with Partial Information

Finally, we briefly mention an even harder setup for online learning, which will be our focus near the end of this course. The difficulty of this setting lies in the fact that the learner only has partial information. Specifically, recall that in the last section, we assume that z_t is revealed to the learner at the end of each round. What if instead the learner only observes partial information of z_t? Is learning still possible?

Multi-armed Bandits. As an example, consider a problem instance where $\mathcal{D} = \mathcal{F} = \{1, \dots, K\}$, $\mathcal{Z} = [0,1]^K$, and $\ell(\widehat{y}, z) = z(\widehat{y})$. In words, each time the learner needs to select one out of K items, denoted by $\widehat{y}_t$, while simultaneously the environment decides the loss of picking each item by specifying a loss vector z_t. The loss of the learner is simply the loss of the selected item, denoted as $z_t(\widehat{y}_t)$. Importantly, instead of revealing the entire vector z_t to the learner, let us consider a harder setting where only the value $z_t(\widehat{y}_t)$ is observed by the learner. This is in fact nothing but the well-known *Multi-armed Bandits* (MAB) problem.

Such a learning formulation has many applications in real-world. For instance, a recommendation system can be naturally cast as an instance of MAB, where the K items correspond to a set of movies, products, or news articles, and selecting an item corresponds to recommending it to the user. Afterwards, the system observes some feedback on the recommendation, which can then be encoded as some loss or reward (e.g. if the user watches the recommended movie, the loss is 0, and otherwise the loss is 1). Importantly, the system does not observe the loss for the items that weren't recommended, which matches the partial information aspect of MAB.

MAB is just one canonical example of online learning with partial information, and we will discuss several more in the future, including general partial monitoring and reinforcement learning. While one can still formally define the value of such games, similar to Equation (1), it is in fact much harder to actually "solve" such a complicated minimax problem, and in particular, there is no obvious way to write it as a sequence of minimax expressions as in Equation (2) (you should try!). Nevertheless, we will still discuss how to design no-regret algorithms for these problems.