# CS229M/STATS214: Machine Learning Theory

Lecturer: Tengyu Ma
Scribe: Tianyu Du, Xin Lu and Soham Sinha

Lecture # 15
Mar 8th, 2021

## 15.1 Review and overview

In the last lecture, we finished off our discussion of algorithmic regularization. In a simplified setting, we proved that gradient descent on the empirical loss $\widehat{L}(\beta)$ with a bounded number of iterations gives a solution $\widehat{\beta}$ that is close to the true solution $\beta^\star$.

In this lecture, we switch gears and talk about *online learning* and *online convex optimization*, which will be the focus of the next three lectures. (The last lecture of this course will cover *generalization guarantees*.) The main idea driving behind online learning is that we move away from the assumption that the training and test data are both drawn i.i.d from some fixed distribution. In the online setting, training data and test data come to the user in an interwoven manner, and data can be generated *adversarially*.

We also describe how online learning can be reduced to online convex optimization and batch learning, as well as the "Follow-the-Leader" algorithm, a simple algorithm for online learning problems.

## 15.2 Online learning

In classical supervised learning, we train the model with the assumption that $(x^{(i)}, y^{(i)}) \overset{i.i.d.}{\sim} P_{\text{train}}$, where $P_{\text{train}}$ is the underlying distribution of the training data. In most cases, we assume the test data, i.e., the data we want our model to predict well, comes from the same distribution (or at least one that is close to $P_{\text{train}}$). Reality is often more complicated: data could indeed be generated in sequence, or even in an adversarial manner, so it is often the case that $P_{\text{test}}$ differs from $P_{\text{train}}$. The situation where $P_{\text{test}}$ and $P_{\text{train}}$ are different is known as *domain shift*. There are some theories that tackle the issue of domain shift and generalization properties of transfer learning. However, the field is still largely being developed. (See [BDBC$^+$07], for example.)

Online learning is an attempt to deal with domain shift in a way that is agnostic to the relationship between the training and test data distributions (i.e. deal with "worst-case" domain shift). As an example, many recommendation systems today collect users' historical trace of shopping behavior, which are not i.i.d. samples, and makes adaptive recommendations based on users' changing shopping behavior. Hence, one can see that online learning attempts to adapt to the constantly evolving reality on time. Notice that unlike the "offline model" (i.e., classical supervised learning), online learning learns while testing, and hence there is no rigid division in time to differentiate training and testing phase.

Online learning has several distinctive features [Lia16]:

1. The data may be *adversarial*. We cannot assume that sample is drawn independently from some distribution.

2. The data and predictions are *sequential*. At each step, the algorithm makes a prediction after given a single piece of data.

3. The feedback is *limited*. For example, in bandit problems, the algorithm only knows if its right or wrong, but no other feedback is given.

### 15.2.1 Setup

Online learning can be viewed as a game between two parties: (i) the learner/agent/algorithm/player, and (ii) the environment/nature. For simplicity, we will refer to the two parties as "learner" and "environment" in the remainder of this lecture.

The game takes place over $T$ rounds or time steps. At each step $t = 1, \dots, T$, the learner receives an input $x_t \in \mathcal{X}$ from the environment and makes a prediction $\hat{y} \in \mathcal{Y}$ in response. The learner then receives the label $y_t$ from the environment and suffers some loss. This procedure is outlined in Algorithm 1 and is illustrated in Figure 15.1.

**Algorithm 1:** General online learning problem

```
1 for $t = 1, ...T$ do
2     Learner receives $x_t \in \mathcal{X}$ from environment, which may be chosen adversarially;
3     Learner predicts $\hat{y} \in \mathcal{Y}$;
4     Learner receives the label $y_t$, from environment, which may be chosen adversarially;
       Learner suffers some loss $\ell(y_t, \hat{y}_t)$.
```

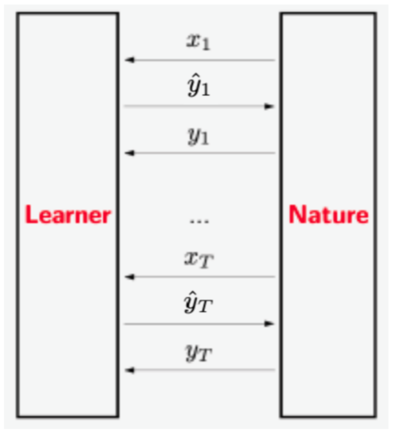

Figure 15.1: A representation of the online learning problem.

Later, we will see that the manner in which nature generates $(x_t, y_t)$ leads to different types of online learning. In the most adversarial setting of online learning, it is possible that the "true label" $y_t$ is not generated at the same time as $x_t$. The environment could generate the label $y_t$ depending on the prediction $\hat{y}_t$ made by the learner. We can also see that Algorithm 1 is a very general framework as there are very few constraints on how $x_t$ and $y_t$ are generated.

### 15.2.2 Evaluation of the learner

Given this setup, a natural question to ask is how one can evaluate the performance of the learner. Intuitively, one could simply evaluate the learner's performance by computing the loss between the predicted label and the "true" label sent by the environment $\ell(y_t, \hat{y}_t)$. For the entire sequence of tasks, one can then evaluate in terms of the cumulative loss:

$$\sum_{t=1}^{T} \ell(y_t, \hat{y}_t). \tag{15.1}$$

However, as the environment can be adversarial, the task itself might be inherently hard and even the best possible learner fails to achieve a small loss. Hence, instead instead of using the cumulative loss for a learner by itself, we compare its performance against a suitable baseline, the "best model in hindsight". Assume that our learner comes from a set of hypotheses $\mathcal{H}$. Let us choose the hypothesis $h \in \mathcal{H}$ that minimizes the cumulative loss, i.e.

$$h^\star = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t)). \tag{15.2}$$

Note here that in minimizing the cumulative loss, the learner gets to see all the data points $(x_t, y_t)$ at once. The cumulative loss of $h^\star$ is the best we can ever hope to do, and so it would be better to compare the cumulative loss of the learner against it. (This approach is analogous to "excess risk", which tells how far the current model is away from the best we could hope for.) This measurement is denoted as *regret*, and is formally defined as:

$$\text{Regret} \triangleq \left[\sum_{t=1}^{T} \ell(y_t, \hat{y}_t)\right] - \underbrace{\left[\min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))\right]}_{\text{best loss in hindsight}} \tag{15.3}$$

Using this definition, if the best model in hindsight performs well, then the learner has more responsibility to learn to predict well in order to match up the performance of the baseline.

### 15.2.3 The realizable case

In general, if the environment is too powerful, leading the learner to a large loss, it will also hinder the best model in hindsight from doing well. On the other hand, there are settings where some members of the hypothesis class can actually do well. Such settings/problems are usually referred to as *realizable*:

**Definition 15.1** (Realizable problem)**.** An online learning problem is *realizable* (for a family of predictors $\mathcal{H}$) if there exists $h \in \mathcal{H}$ such that for any $T$, $\sum_{t=1}^{T} \ell(y_t, h(x_t)) = 0$.

Note that even though zero error is possible, this is still an interesting problem to consider because the $x_t$'s are not i.i.d. as they are in classical supervised learning. Hence, standard statistical learning theory does not apply, and there is still research to be done here.

**Example 15.2.** Consider a classification problem on $(x_t, y_t)$, and for simplicity assume $y_t \in \{0, 1\}$. Suppose there exists $h^\star \in \mathcal{H}$ such that we always have $y_t = \hat{y}_t^\star = h^\star(x_t)$. In this case, the problem is realizable.

In this case, the learner can adopt a "majority algorithm". At each time, the learner maintains a set $V_t \subset \mathcal{H}$ so that $\sum_{t=1}^{T} \ell(y_t, h(x_t)) = 0$ for all $h \in V_t$, and $\hat{y}_t$ is simply the prediction made by the majority of $h \in V_t$. Based on the loss received, learners $h \in V_t$ that fail for time $t+1$ will be eliminated from future $V_t$'s.

With this setup, we can see that for each wrong prediction made by the learner, at least half of the hypotheses $h \in V_t$ will be eliminated. Hence, $1 \leq |V_{t+1}| \leq |\mathcal{H}| 2^{-M}$ where $M$ is the number of mistakes made so far. Thus, one has $M \leq \log |\mathcal{H}|$ by taking log on both sides of inequalities and rearrange.

Now, if one puts $\ell$ as the zero-one loss, the regret for this example will be

$$\text{Regret} = \sum_{t=1}^{T} \ell(y_t, h(x_t)) = M, \tag{15.4}$$

so in this example, one has regret $\leq \log |\mathcal{H}|$, which is a non-trivial bound when $\mathcal{H}$ is finite.

As one can see in the example, the realizable case usually indicates that the problem is not too far out of reach. Indeed, for finite hypothesis classes and linear models, the realizable case is considered to be straightforward to solve. This is perhaps why most of the past literature has focused on non-realizable cases. However, the realizable case is still an interesting problem and perhaps a very good starting point when the model class is beyond linear models and when the loss function is no longer convex, because the $x_t$'s are not i.i.d. as they are in classical supervised learning. Hence, standard statistical learning theory does not apply, and there is still research to be done here.

In the rest of the lecture, we will only focus on the convex loss case, where we reduce online learning to online convex optimization.

## 15.3 Online (convex) optimization (OCO)

*Online convex optimization (OCO)* is a particularly useful tool to get results for online learning. Many online learning problems (and many other types of problems!) can be reduced to OCO problems, which allow them to be solved and analyzed algorithmically. Algorithm 2 describes the OCO problem, which is more general than the online learning problem. (Note: *Online optimization (OO)* refers to Algorithm 2 except that the $f_t$'s need not be convex. However, due to the difficulty in non-convex function optimization, most research has focused on OCO.)

**Algorithm 2:** Online (convex) optimization problem

```
1 for $t = 1, ..., T$ do
2     The learner picks some action $w_t \in \Omega$ from the action space $\Omega$;
3     The environment picks a (convex) function $f_t : \Omega \to [0,1]$;
4     The learner suffers the loss $f_t(w_t)$ and observes the entire loss function $f_t(\cdot)$.
```

Essentially the learner is trying to minimize the function $f_t$ at each step. As with online learning, one evaluates the performance of learner in online optimization setting using the regret:

$$\text{Regret} = \sum_{t=1}^{T} f_t(w_t) - \underbrace{\min_{w \in \Omega} \sum_{t=1}^{T} f_t(w)}_{\text{best action in hindsight}} . \tag{15.5}$$

At some level, OCO seems like an impossible task, since we are trying to minimize a function $f_t$ that we only get to see *after* we have made our prediction! This is certainly the case for $t = 1$. However, as time goes on, we see more and more functions and, if future functions are somewhat related to past functions, we have more information to make better predictions. (And if the future functions are completely unrelated or contradictory to past functions, then the best action in hindsight would also be bad and therefore our algorithm does not have to do much.)

### 15.3.1 Settings and variants of OCO

There are multiple settings of the OCO network, which can vary the power of the environment and observations.

- Stochastic setting: $f_1, ..., f_T$ are i.i.d samples from some distribution $P$. This corresponds to $(x_t, y_t)$ being i.i.d. in online learning. Under this setting, the environment is not adversarial.
- Oblivious setting: $f_1, ..., f_T$ are chosen arbitrarily but before the game starts. This corresponds to $(x_t, y_t$ being chosen before the game starts. In this setting, the environment can be adversarial but cannot be adaptive. The environment can choose these functions based on the learner's algorithm, but not the actual action if the learner's algorithm contains randomness. (This is the setting that we focus on in this course.)
- Non-oblivious/adaptive setting: For all $t$, $f_t$ can depend on the learner's actions $w_1, ...w_t$. Under this setting, the environment can be adversarial and adaptive. This is the most challenging setting because the environment is powerful enough to know not only the strategy of the learner, but also the exact choice the learner finally made. (Note however that If the learner is deterministic, the environment does not have more power here than in the oblivious setting. The oblivious adversary can simulate the game before the game starts, and chose the most adversarial input accordingly.)

## 15.4 Reducing online learning to online optimization

There is a natural way to reduce the online learning problem to online optimization, with respect to a specific type of model $h_w$ parametrized by $w \in \Omega$. Recall that in online learning problem, the learner predicts $y_t$ upon receiving $x_t$. If the learner possesses oracle to solve online optimization problem, the learner can consult the oracle to obtain $w_t$, the parameter of the model as in online optimization problem, and then predict $\hat{y}_t = h_{w_t}(x_t)$.

In the next two subsections, we give two examples of how an online learning problem can be reduced to an OCO problem.

### 15.4.1 Example: Online learning regression problem

Consider the regression model $h_w(x) = w^\top x$ parameterized by $w$ in parameter space $\Omega$ with squared error loss $\ell$. Here is the online learning formulation of the regression problem:

This can be reduced to the OCO problem in the following way:

In this example, we have the following correspondence:

- $f_t$ in online optimization $\leftrightarrow$ squared error loss functions for $(x_t, y_t)$.

**Algorithm 3:** Online learning regression problem

```
1 for $t = 1, ..., T$ do
2     The learner receives $x_t \in \mathbb{R}^d$ from the environment;
3     The learner predicts $\hat{y}_t$;
4     The environment selects $y_t$ and sends it to the learner;
5     The learner suffers loss $\ell(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2$.
```

**Algorithm 4:** OCO formulation of regression problem

```
1 for $t = 1, ..., T$ do
2     The learner receives $x_t \in \mathbb{R}^d$ from the environment;
3     The learner gives $x_t$ to the OCO solver and obtains $w_t \in \mathbb{R}^d$;
4     The learner predicts $\hat{y}_t = h_{w_t}(x_t) = w_t^\top x_t$;
5     The environment selects $y_t$ and sends it to the learner;
6     The learner suffers loss $(y_t - h_{w_t}(x_t))^2$;
7     With $(x_t, y_t)$ observed, the learner can reconstruct the loss function
        $f_t(w) = (y_t - h_w(x_t))^2$ and give it to the OCO solver.
```

- $w_t$ in online optimization $\leftrightarrow$ parameters of the linear model $h_{w_t}$.

Since $h_w(\cdot)$ is linear, the corresponding squared error loss function $f_t$ are convex, and so we have effectively reduced the online linear regression problem to an online *convex* optimization problem.

Notice that in the previous example, the loss function $f_t$ actually depends on the label $y_t$, which demonstrates that the key challenge in online optimization is that the function $f_t$ is unknown to the learner when the prediction $\hat{y}_t$ is made.

### 15.4.2 Example: The expert problem

Suppose we wish to predict tomorrow's weather and 10 different TV channels provide different forecasts. Which one should we follow? Formally, consider a finite hypothesis class $\mathcal{H}$, where each $h \in \mathcal{H}$ represents an expert, and we wish to choose a $h_t$ wisely at each time step. For simplicity, we assume the prediction is binary, i.e. $\hat{y} \in \{0, 1\}$, and suppose the loss function is 0-1 loss. (The problem can easily be generalized to more general predictions and losses.) The problem is outlined in Algorithm 5.

**Algorithm 5:** The expert problem

```
1 for $t = 1, ..., T$ do
2     The learner obtains predictions from $N$ experts;
3     The learner chooses to follow prediction of one of the experts $i_t \in [N]$;
4     The environment gives the learner the true value. The learner is thus able to learn the
        loss of each of the experts: $\ell_t \in \{0, 1\}^N$;
5     The learner suffers the loss of the expert which was chosen: $\ell_t(i_t)$.
```

We want to design a method that chooses $i_t$ for each step (line 3 in Algorithm 5) to minimize the regret:

$$\text{Regret} \overset{\Delta}{=} \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(i_t) - \underbrace{\min_{i\in[N]} \sum_{t=1}^{T} \ell_t(i)}_{\text{the best expert in hindsight}}\right], \tag{15.6}$$

where the expected value is over $i_t$, thus covering the case where the $i_t$'s could be random.

To make the expert problem amenable to reduction to OCO, we introduce idea of a *continuous action space*. Instead of choosing $i_t$ from $\Omega = [N]$, the learner chooses a distribution $p_t$ from the $N$-dimensional simplex $\Delta(N) = \left\{p \in \mathbb{R}^N : \|p\|_1 = 1, p \geq 0\right\}$. The learner then samples $i_t \sim p_t$. With this formulation, instead of selecting particular expert $i_t$ to follow, the learner adjusts the belief $p_t$, and samples from the distribution to choose which expert to follow. Algorithm 6 outlines this procedure. Note that the loss is the expected loss $\mathbb{E}_{i\sim p_t}[\ell_t(i)]$ instead of the sampled $\ell_t(i_t)$.

**Algorithm 6:** The expert problem with continuous action

```
1 for $t = 1, ..., T$ do
2     The learner obtains predictions from $N$ experts;
3     The learner chooses a distribution $p_t \in \Delta(N)$;
4     The learner samples one expert $i_t \sim p_t$;
5     The environment gives the learner the true value and the loss/error of all experts:
        $\ell_t \in \{0,1\}^N$;
6     The learner suffers expected loss $\sum_{i\in[N]} p_t(i)\ell_t(i) = \langle p_t, \ell_t\rangle$;
```

With the continuous action space, it is easy to reduce the expert problem to an OCO: see Algorithm 7. (The problem is convex since the loss function is convex and the parameter space $\Delta(N)$ is convex.)

**Algorithm 7:** The expert problem

```
1 for $t = 1, ..., T$ do
2     The learner obtains predictions from $N$ experts;
3     The learner invokes the OCO oracle to obtain $p_t \in \Delta(N)$;
4     The learner chooses to follow prediction of one of the experts $i_t \in [N]$;
5     The environment gives the learner the true value. The learner is thus able to learn the
        loss of each of the experts: $\ell_t \in \{0,1\}^N$;
6     The learner suffers the loss of the expert which was chosen: $\ell_t(i_t)$. The learner can
        reconstruct the loss function $f_t(p) = \langle p, \ell_t\rangle$ and give it to the OCO oracle.
```

In this setting, one can rewrite the regret as:

$$\text{Regret} = \sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{i \in [N]} \sum_{t=1}^{T} \ell_t(i) \tag{15.7}$$

$$= \sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{p \in \Delta(N)} \sum_{t=1}^{T} \langle p, \ell_t \rangle \tag{15.8}$$

$$= \sum_{t=1}^{T} f_t(p_t) - \min_{p \in \Delta(N)} \sum_{t=1}^{T} f_t(p). \tag{15.9}$$

We obtain (15.8) because

$$\sum_{t=1}^{T} \langle p, \ell_t \rangle = \left\langle p, \sum_{t=1}^{T} \ell_t \right\rangle \geq \min_{i \in [N]} \left[ \sum_{t=1}^{T} \ell_t(i) \right], \tag{15.10}$$

with equality for the probability distribution $p(i) = 1$ when $i = \operatorname{argmin}_i \left[ \sum_{t=1}^{T} \ell_t(i) \right]$ and $p(i) = 0$ otherwise, and (15.9) is by definition of $f_t$.

## 15.5 Reducing online learning to batch learning

In this section, we present a reduction from online learning to standard supervised learning problem, also known as the "batch problem" in this literature.

As in the standard supervised learning setting, consider an i.i.d dataset $\{(x_t, y_t)\}_{t=1}^{T}$ and some parameter $w$. Let $L(w)$ and $\widehat{L}(w)$ be the population loss and empirical loss respectively. For simplicity, assume $|\ell((x_i, y_i), w)| \leq 1$. The theorem below establishes a link between the regret obtained in online learning and the excess risk obtained in the batch setting.

**Theorem 15.3** (Relationship between excess risk and regret)**.** *Assume $\ell((x, y), w)$ is convex. Suppose we run an online learning algorithm on the dataset $\{(x_i, y_i)\}_{i=1}^{T}$ and obtain a sequence of models $w_1, \ldots, w_T$, and regret $R_T$. Let $\overline{w} = \frac{1}{T} \sum_{i=1}^{T} w_i$, then the excess risk of $\overline{w}$ can be bounded above:*

$$L(\overline{w}) - L(w^\star) \leq \frac{R_T}{T} + \widetilde{O}\left( \frac{1}{\sqrt{T}} \right), \tag{15.11}$$

*where $w^\star = \operatorname{argmin}_{w \in \Omega} L(w)$.*

Here are some intuitive interpretations of the theorem:

- If $R_T = O(T)$, then we have some non-trivial result. Otherwise, the bound in (15.11) is increasing $T$ and does not provide any useful information.
- If the batch problem has a $1/\sqrt{T}$ generalization bound, then the best you can hope for in online learning is $R_T = O(\sqrt{T})$.
- If the batch problem has a $1/T$ generalization bound, you can hope for $O(1)$ regret (or $\widetilde{O}(1)$ regret in some cases).
- We often have $O(\sqrt{T})$ excess risk supervised learning problems; hence it is reasonable to expect $O(\sqrt{T})$ regret in online learning problems.

## 15.6 Follow-the-Leader (FTL) algorithm

In this section, we analyze an algorithm called "Follow-the-Leader" (FTL) for OCO, which is intuitive but fails to perform well in many cases.

The FTL algorithm behaves as its name suggests: it always selects the action $w_t$ such that it minimizes the historical loss the learner has seen so far, i.e.

$$w_t = \operatorname*{argmin}_{w \in \Omega} \sum_{i=1}^{t-1} f_i(w). \tag{15.12}$$

We now demonstrate how the FTL algorithm can fail for the expert problem. In the expert problem, $f_t(p) = \langle p, \ell_t \rangle$, so

$$\begin{aligned} p_t &= \operatorname*{argmin}_{p \in \Delta(N)} \sum_{i=1}^{t-1} f_i(p) && (15.13)\\ &= \operatorname*{argmin}_{p \in \Delta(N)} \sum_{i=1}^{t-1} \langle \ell_i, p \rangle && (15.14)\\ &= \operatorname*{argmin}_{p \in \Delta(N)} \left\langle \sum_{i=1}^{t-1} \ell_i, p \right\rangle. && (15.15) \end{aligned}$$

The minimizer $p \in \Delta(N)$ is a point-mass probability, with the point mass at the smallest coordinate of $\sum_{i=1}^{t-1} \ell_i$. This gives regret

$$\text{Regret} = \sum_{i=1}^{t-1} \ell_i(i_t), \quad \text{where } i_t = \operatorname*{argmin}_{j \in [N]} \sum_{i=1}^{t-1} \ell_i(j). \tag{15.16}$$

Now, consider the following example: suppose we have only two experts. Suppose expert 1 makes perfect predictions on even days while expert 2 makes perfect predictions on odd days. Assume also that the FTL algorithm chooses expert 1 to break ties (this is not an important point but makes the exposition simpler.) In this setting, the FTL algorithm always selects the *wrong* expert to follow. A few rounds of simulation of this example is shown in Table 15.1.

Table 15.1: An example where FTL fails

| Day | 1 | 2 | 3 | 4 | ... | ... |
|---|---|---|---|---|---|---|
| Expert 1's loss | 1 | 0 | 1 | 0 | ... | ... |
| Expert 2's loss | 0 | 1 | 0 | 1 | ... | ... |
| FTL choice $i_t$ | 1 | 2 | 1 | 2 | 1 | ... |

The best expert in hindsight has a loss of $T/2$ (choosing either expert all the time incurs this loss, and so the regret of the FTL algorithm is $T - T/2 = T/2 = \Theta(T)$. The main reason for FTL's failure is that is a deterministic algorithm driven by an extreme update, with no consideration on potential domain shift (it always selects the best expert based on the past with no consideration of the potential next $f_t$). Knowing its deterministic strategy, the environment can easily play in an adversarial manner. To perform better in a problem like this, we need some randomness to hedge risk.

# Bibliography

[BDBC$^+$07] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al., *Analysis of representations for domain adaptation*, Advances in neural information processing systems **19** (2007), 137.

[Lia16] Percy Liang, *Cs229t/stat231: Statistical learning theory (winter 2016)*, April 2016.