CS229T/STATS231: Statistical Learning Theory

Lecturer: Tengyu Ma
Scribe: Shiori Sagawa

Lecture 14
November 7, 2018

1 Review and Overview

In the last lecture, we introduced online learning. In online learning, a learner and environment interact as follows. In each iteration $t = 1, ..., T$:

- Learner receives an input $x_t \in \mathcal{X}$ from the environment
- Learner predicts $\hat{y}_t \in \mathcal{Y}$, based on $x_{1:t}, y_{1:t-1}$
- Learner receives the true label $y_t \in \mathcal{Y}$ from the environment
- Learner suffers loss $\ell(y_t, \hat{y}_t)$

To evaluate a learner, we compare against an optimal mapping $h : \mathcal{X} \mapsto \mathcal{Y}$. More concretely, we evaluate through regret:

$$\text{regret} \triangleq \sum_{t=1}^{T} \ell(y_t, \hat{y}_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))$$

In this lecture, we introduce online convex optimization (OCO). We show that many online learning problems can be reduced to this framework, showing that we can solve such online learning problems by solving OCO. This motivates us to study algorithms for OCO. We then try out an intuitive algorithm for OCO called Follow the Leader (FTL), show that the algorithm can perform poorly through an example, and gain intuition on some features OCO algorithms should have.

2 Online Convex Optimization Framework

Below is the set-up of online convex optimization.

In each iteration $t = 1...T$:

- Player picks an action $\omega_t \in \Omega$, where Ω is a convex set
- Environment picks a convex function $f_t : \Omega \mapsto [0, 1]$
- Player observes some information about f_t

We optimize w_t to minimize regret:

$$\text{regret} \triangleq \sum_{t=1}^{T} f_t(\omega_t) - \min_{\omega \in \Omega} \sum_{t=1}^{T} f_t(\omega)$$

3 Reduction of Online Learning Problems to OCO

In this section, we motivate our focus on OCO. Many interesting online learning problems can be reduced to OCO, and such problems can be solved by solving OCO.

3.1 Online Linear Regression

Online learning formulation:

In each iteration $t = 1...T$:

- Player receives $x_t \in \mathbb{R}^d$
- Environment selects y_t
- Player predicts $\hat{y}_t$
- Player receives y_t from the environment
- Player suffers loss $\ell(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2$

Online convex optimization formulation:

In each iteration $t = 1, ..., T$:

- Player receives $x_t \in \mathbb{R}^d$
- Environment selects y_t.
  Equivalently, the environment specifies $f_t(w) = \ell(y_t, w^T x_t) = (y_t - w^T x_t)^2$.
- Player calls OCO to select $w_t \in \Omega = \mathbb{R}^d$, in turn predicting $\hat{y}_t = w_t^T x_t$
- Player observes y_t, which is equivalent to observing f_t.

Regret

Because f_t in OCO is the loss function for the data at iteration t, regrets for online learning and OCO problems are equivalent.

$$\begin{aligned}\sum_{t=1}^{T} f_t(w_t) - \min_{w \in \Omega} \sum_{t=1}^{T} f_t(w) &= \sum_{t=1}^{T} \ell(y_t, w_t^T x_t) - \min_{w \in \mathbb{R}^d} \sum_{t=1}^{T} \ell(y_t, w^T x_t) \\ &= \sum_{t=1}^{T} \ell(y_t, \hat{y}_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))\end{aligned}$$

3.2 Expert Problem

Online learning formulation:

In each iteration $t = 1...T$:

- Player receives predictions from N experts from the environment
- Environment specifies N expert predictions' losses $\ell_t \in [0,1]^N$, where $\ell_t[i]$ is the loss of the expert i
- Player picks one expert to follow by drawing randomly from discrete distribution $p_t \in \{\Delta(N)\}$, where $\{\Delta(N)\} \triangleq \{p \in \mathbb{R}^N : ||p||_1 = 1, p \geq 0\}$. In other words, $\{\Delta(N)\}$ is a set of all probability vectors in $\mathbb{R}^N$, where ith item ($p_t[i]$) is the probability of selecting expert i
- Player observes ℓ_t
- Player suffers loss $\mathbb{E}_{i \sim p_t}[\ell_t[i]] = \langle \ell_t, p_t \rangle$

Online convex optimization formulation:

In each iteration $t = 1, ..., T$:

- Player receives predictions from N experts from the environment
- Environment specifies ℓ_t. Equivalently, the environment specifies $f_t(p) = \langle \ell_t, p \rangle$
- Player calls OCO to pick $p_t \in \Omega$, where $\Omega = \{\Delta(N)\}$. Once p_t is selected, the player draws an expert from the distribution p_t and follows the drawn expert.
- Player observes ℓ_t, which is equivalent to observing f_t

Regret

Because f_t in OCO is the loss function for the data at iteration t, regrets for online learning and OCO problems are equivalent.

$$\begin{aligned}\sum_{t=1}^{T} f_t(p_t) - \min_{p \in \Omega} \sum_{t=1}^{T} f_t(p) &= \sum_{t=1}^{T} \langle \ell_t, p_t \rangle - \min_{p \in \{\Delta(N)\}} \sum_{t=1}^{T} \langle \ell_t, p \rangle \\ &= \sum_{t=1}^{T} \langle \ell_t, p_t \rangle - \min_{i \in \{1,...,N\}} \sum_{t=1}^{T} \ell_t[i] \qquad \text{(solution to a linear program is a vertex)}\end{aligned}$$

3.3 Sequential Investment

Set-Up:

Starting with capital W_1 in the first iteration, invest all of your capital across N assets and collect returns at each iteration.

Online learning formulation:

In each iteration $t = 1...T$:

- Player selects $p_t \in \{\Delta N\}$ and invests $W_t p_t[i]$ to asset i for each i
- Environment specifies the multiplicative return $\gamma_t \in \mathbb{R}^N$
- Player observes γ_t.
- Player's capital is updated as $W_{t+1} = \sum_{i=1}^{N} W_t p_t[i]\gamma_t[i]$.
  Player suffers loss $\log(\frac{W_t}{W_{t+1}}) = -\log(\sum_{i=1}^{N} p_t[i]\gamma_t[i]) = -\log\langle p_t, \gamma_t\rangle$

Online convex optimization formulation:

In each iteration $t = 1, ..., T$:

- Player calls OCO to select $p_t \in \{\Delta N\}$ and invests $W_t p_t[i]$ to asset i for each i
- Environment specifies the multiplicative return γ_t. Equivalently, the environment specifies $f_t(p) = \langle p, \gamma_t\rangle$.
- Player observes γ_t, which is equivalent to observing f_t.

Regret

Because f_t in OCO is the loss function for the data at iteration t, regrets for online learning and OCO problems are equivalent.

$$\begin{aligned}\sum_{t=1}^{T} f_t(p_t) - \min_{p\in\Omega} \sum_{t=1}^{T} f_t(p) &= \sum_{t=1}^{T} \langle \gamma_t, p_t\rangle - \min_{p\in\{\Delta(N)\}} \sum_{t=1}^{T} \langle \gamma_t, p\rangle \\ &= \sum_{t=1}^{T} \langle \gamma_t, p_t\rangle - \min_{i\in\{1,...,N\}} \sum_{t=1}^{T} \gamma_t[i] \qquad \text{(solution to a linear program is a vertex)}\end{aligned}$$

4 Settings and Variants of OCO Framework

There are multiple settings of the OCO framework. It can vary in the power of environment and observations.

Power of Environment
The environment can choose the f_t with different levels of adversarialism and adaptivity.

- Stochastic setting: $f_1, ..., f_T$ are i.i.d. samples from some distribution P. Under this setting, the environment is not adversarial.
- Oblivious setting: $f_1, ..., f_T$ are selected before the game starts. Under this setting, the environment can be adversarial, but cannot be adaptive.
- Non-oblivious/adaptive setting: $\forall t, f_t$ can depend on $w_1, ..., w_t$. Under this setting, the environment can be adversarial and adaptive. Note that if the learner is deterministic, then the environment doesn't get more advantages moving from a oblivious setting to a non-oblivious setting.

Power of Observations
Player can observe different amounts of information on f_t.

- Full-information setting: player observes the entire f_t
- Bandit setting: player only observes $f_t(w_t)$

In the rest of the lecture, we focus on a oblivious, full-information setting.

5 Comparison of Online Learning and Batch Supervised Learning

Consider a batch supervised learning problem with convex loss function $\ell(\omega, z)$, where ω is a parameter and z is a labeled data point. This problem can be reduced to OCO. In each iteration, the environment specifies $f_t(\omega) = \ell(\omega, z_t)$ (i.e. the environment is stochastic since $z_1, ..., z_T$ are drawn i.i.d. from P_z), and the player calls OCO to select a parameter ω_t and experiences loss $f_t(\omega_t)$. After all T iterations, the learner aggregates w_t into one parameter $\bar{\omega}$, for example by taking an average of w_t. The excess risk of such online algorithm can be bounded by the regret.

Theorem (informal):
Let $\bar{\omega}$ be the parameter selected by the above online learning algorithm and let $L(\omega) = \mathbb{E}[\ell(\omega, z)]$ be the expected loss for ω.

$$L(\bar{\omega}) - L(\omega^*) \leq \frac{\text{regret}}{T}$$

There are two key takeaways from the reduction and the above theorem.

1. Online learning is harder than batch supervised learning.
2. We now have a sense of what regret bound to aim for. Because the excess risk is typically bounded by $O(\frac{1}{\sqrt{T}})$, the regret should be able to be bounded by $O(\sqrt{T})$.

6 Follow the Leader (FTL) algorithm

So far, we have shown that solving OCO is sufficient to solve many online learning problems. We now will be designing algorithms for OCO, particularly for the oblivious setting. As a starting point, we will try out an intuitive algorithm called the Follow the Leader (FTL) algorithm. As we show that the algorithm can perform poorly through an example, we will get an intuition for how we can improve it.

Follow the Leader (FTL) Algorithm
At high level, FTL selects an action ω_t that minimizes the cumulative loss so far (in iterations $1, ..., t-1$). More concretely, at each iteration t, we select the following ω_t:

$$\omega_t = \text{argmin}_{\omega \in \Omega} \sum_{i=1}^{t-1} f_i(\omega)$$

Example: Expert Problem
Applying the FTL algorithm to the expert problem (see section 3.2), the learner selects an expert j_t that has the smallest cumulative loss so far.

$$\begin{aligned} p_t &= \text{argmin}_{p \in \Omega} \sum_{i=1}^{t-1} f_i(p) \\ &= \text{argmin}_{p \in \Omega} \sum_{i=1}^{t-1} \langle \ell_t, p \rangle \\ j_t &= \text{argmin}_{j \in \{1,...,N\}} \sum_{i=1}^{t-1} \ell_t[j] \qquad \text{(solution to a linear program is a vertex)} \end{aligned}$$

In cases when the FTL algorithm doesn't specify the leader to follow (i.e. at $t = 1$ and when there are ties), the learner picks an expert with the lowest index.

We now show that the above algorithm can perform poorly, showing that there is a sequence of losses where the worst-case regret is obtained. Let $N = 2$ and design the ℓ_t as follows:

$$\ell_t = \begin{cases} 1 & t \text{ is odd} \\ 0 & t \text{ is even} \end{cases}$$

Following the FTL algorithm, we would get the following p_t:

$$p_t = \begin{cases} 1 & t \text{ is odd} \\ 0 & t \text{ is even} \end{cases}$$

	ℓ_1	ℓ_2	ℓ_3	...	ℓ_t
Expert 1	1	0	1		
Expert 2	0	1	0		

	p_1	p_2	p_3	...	p_t
Expert 1	1	0	1		
Expert 2	0	1	0		

Computing the regret under this setting:

$$\begin{aligned}
\text{regret} &= \sum_{t=1}^{T} f_t(p_t) - \min_{p \in \Omega} \sum_{t=1}^{T} f_t(p) \\
&= \sum_{t=1}^{T} \langle \ell_t, p_t \rangle - \min_{i \in \{1,\ldots,N\}} \sum_{t=1}^{T} \ell_t[i] \\
&= \sum_{t=1}^{T} 1 - \min_{i \in \{1,\ldots,N\}} \sum_{t=1}^{T} \ell_t[1] \\
&= \frac{T}{2}
\end{aligned}$$

The algorithm is performing poorly, getting the worst possible regret. We also notice that the regret is worse than our goal for the regret bound, which is $O(\sqrt{T})$.

Improving the Algorithm

How can we improve the algorithm? We identify two shortcomings of the FTL algorithm and propose to fix them.

1. If the algorithm is deterministic, then the environment can simulate the algorithm and provide adversarial f_t. Randomness should be added to the algorithm.

2. We saw that frequently switching the selected expert hurt the performance. The algorithm should be more conservative about switching.