1 Outline

In this lecture, we cover

- sequential decision making problems,
- online convex optimization,
- algorithms for online convex optimization.

2 Sequential Decision Making Problems

In this lecture, we discuss the framework of *online convex optimization (OCO)*. Online convex optimization (OCO) is an online learning problem, that is to make a sequence of predictions based on the history of past decisions and their results. The framework of OCO is closely related to game theory, statistical learning theory, and stochastic modeling as well as convex optimization. Before we formally define the OCO framework, let us first discuss several sequential decision-making problems.

- (Online spam filtering) We receive emails repeatedly, for each of which we apply an existing spam-filtering system. A spam-filtering system has a list of words and expressions, based on which it can predict whether an email is spam or valid. When an email that is classified as valid by the existing filter turns out to be spam, we have to update the filter so that we can filter similar spam emails later.
- (Online advertisement selection) A web browser selects a collection of online advertisements for its ad slots. The web browser posts a catalog of online ads and observes their popularity from users by the click-through rates. Later, the browser can change its ad selection based on its prediction about user demands.
- (Adversarial multi-armed bandits) A player repeatedly plays a slot machine (or bandit) among multiple slot machines. Each slot machine provides a reward when played, and the player aims to maximize the cumulative reward over time. However, the reward structure of each slot machine may change over time in an adversarial manner.
- (Portfolio management) An investor repeatedly allocates its wealth to multiple assets such as stocks and bonds. The return rates of the assets may change over time in an adversarial manner, and the investor aims to maximize the cumulative return over time.

2.1 Online Binary Classification

Let us consider a mathematical model to establish an email spam filtering system. Recall that we used the support vector machine (SVM) for binary classification. Just to remind you what it was, we find a pair of a coefficient vector w and a right-hand side value b to use the hyperplane $w^\top x = b$ to classify data points. Given a feature vector x, we assign it label $\text{sign}(w^\top x - b)$ where $\text{sign}(c)$ has

value 1 if $c \geq 0$ and value -1 if $c < 0$. When a training set of multiple data is available, we can find such a classifier (w, b) by solving a convex optimization problem whose objective is to minimize the hinge loss.

However, in some scenarios, data points dynamically arrive so that we gradually accumulate the data. In such cases, we may adjust our model over time, and the learning process continues. To be more specific, let us consider the online binary classification problem described as follows. An email is represented by its feature vector $x \in \mathbb{R}^d$ and label $y \in \{-1, 1\}$. The feature vector can encode words and expressions written in it, while the label indicates whether the email is spam or not. Let's say that $y = 1$ indicates spam and $y = -1$ indicates valid. For each time slot t, we repeat the following procedure.

- The spam filtering system prepares a classifier (w_t, b_t) based on the past emails represented by $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1}) \in \mathbb{R}^d \times \{-1, 1\}$.
- New email with feature vector x_t arrives.
- The spam filter predicts that its label is $\mathrm{sign}(w_t^\top x_t - b)$, while the true label of the email is y_t.
- The spam filter incurs a loss of $\max\{0, 1 - y_t(w_t^\top x_t - b)\}$.

After T emails, the cumulative loss is given by

$$\sum_{t=1}^{T} \max\{0, 1 - y_t(w_t^\top x_t - b)\}.$$

Compared to a best classifier, we incur

$$\sum_{t=1}^{T} \max\{0, 1 - y_t(w_t^\top x_t - b)\} - \min_{(w,b)\in\mathbb{R}^d\times\mathbb{R}} \sum_{t=1}^{T} \max\{0, 1 - y_t(w^\top x_t - b)\}$$

more loss. Denoting the loss function at each time t as

$$f_t(w, b) = \max\{0, 1 - y_t(w^\top x_t - b)\},$$

the excess cumulative loss is rewritten as

$$\sum_{t=1}^{T} f_t(w_t, b_t) - \min_{(w,b)\in\mathbb{R}^d\times\mathbb{R}} \sum_{t=1}^{T} f_t(w, b).$$

Therefore, the online binary classification problem is an instance of online convex optimization where the best fixed decision corresponds to the best spam classifier.

2.2 Adversarial Multi-Armed Bandits

Suppose that we have d slot machines (or bandits). Then, at each t, the player chooses which slot machine to play. Here, let $i_t \in \{1, \ldots, d\}$ be the machine that the player chooses at time t. Then the reward of playing machine $i \in \{1, \ldots, d\}$ at time t is given by $r_{i,t}$, which is revealed only after a play. Then we may compare the player's cumulative reward against the total reward of the best slot machine as follows.

$$\max_{i\in\{1,\ldots,d\}} \sum_{t=1}^{T} r_{t,i} - \sum_{t=1}^{T} r_{t,i_t}.$$

2.3 Sequential Investment

Suppose that we have d financial assets such as stocks and bonds. An investor starts with an initial wealth W_1 at time $t = 1$. Then, at each time t, the investor chooses a portfolio vector $x_t \in \{x \in [0,1]^d : \sum_{i=1}^d x_i = 1\}$ where $x_{t,i}$ is the fraction of wealth allocated to asset i at time t. At the end of time t, the return rate of asset i at time t is given by $r_{i,t}$, which is revealed only after the investment. Then, the investor's wealth at time $t+1$ is given by

$$W_{t+1} = W_t \sum_{i=1}^{d} x_{t,i} r_{i,t} = W_t r_t^\top x_t.$$

After T time steps, the investor's wealth is given by

$$W_{T+1} = W_1 \prod_{t=1}^{T} r_t^\top x_t.$$

To maximize W_{T+1}, we may equivalently maximize the logarithm of the wealth:

$$\log W_{T+1} = \log W_1 + \sum_{t=1}^{T} \log \left(r_t^\top x_t \right).$$

Then, we may compare the investor's cumulative log-wealth against that of the best fixed portfolio as follows:

$$\max_{x \in \{x \in [0,1]^d : \sum_{i=1}^d x_i = 1\}} \sum_{t=1}^{T} \log \left(r_t^\top x \right) - \sum_{t=1}^{T} \log \left(r_t^\top x_t \right).$$

3 Regret Minimization for Online Convex Optimization

Online advertisement selection, email spam filter, multi-armed bandits, and sequential investment all involve sequential decision-making proedures that depend on interactions between decisions made by the decision-maker and data provided by the environment. As mentioned earlier, this process is called online learning or online optimization in the sense that the learning process and the optimization task proceed based on the history of information accumulated so far. As opposed to online learning and online optimization, offline learning and offline optimization assume that complete information is available.

The following gives the list of main components.

1. (A sequence of convex loss functions) We are given convex loss functions $f_1, \ldots, f_T$ where T is the length of time horizon. The functions are revealed one at a time sequentially.
2. (Sequential decisions) At each time step t, we get to choose a decision/prediction x_t before the function f_t for the time step is revealed. In other words, the function f_t is unknown to the decision maker when making a decision.
3. (Bounded domain) The set of available decisions (the feasible set), denoted C, is bounded and convex.

Then we compute the accumulated losses incurred over the T time steps.

$$\sum_{t=1}^{T} f_t(x_t).$$

This is indeed an online learning problem because, to make a new decision x_{t+1}, we may use the history of the past decisions and their corresponding losses

$$x_1,\ f_1(x_1),\ x_2,\ f_2(x_2),\ \ldots,\ x_t,\ f_t(x_t)$$

although the loss function f_{t+1} for time step $t+1$ is not yet given.

3.1 Performance Metric: the Notion of Regret

Let $\mathcal{A}$ be an algoriothm for online convex optimization, and let $x_1^{\mathcal{A}}, \ldots, x_T^{\mathcal{A}}$ denote the decisions made by algorithm $\mathcal{A}$. We have defined the cumulative loss, minimizing which is our goal basically. At the same time, to measure how close algorithm $\mathcal{A}$ is to being optimal, we compare the cumulative loss of algorithm $\mathcal{A}$ against the cumulative loss of the best fixed decision. To be more precise, we consider the following notion of *regret*.

$$\text{Regret}_T(\mathcal{A}) = \sum_{t=1}^{T} f_t(x_t^{\mathcal{A}}) - \min_{x \in C} \sum_{t=1}^{T} f_t(x).$$

Here, setting the benchmark as a single best decision is motivated by email spam filter for which we need to find the most effective spam filtering system and multi-armed bandits in which the goal is to find the most profitable slot machine.

We focus on developing algorithms that minimize the regret. By taking a sequence of actions to minimize the regret, we learn and get close to the action of the best decision maker.

Our goal is to design an algorithm $\mathcal{A}$ whose regret is sublinear in T, which means that $\text{Regret}_T(\mathcal{A}) = o(T)$. What does this indicate? We look at the time averaged regret.

$$\frac{1}{T}\sum_{t=1}^{T} f_t(x_t^{\mathcal{A}}) - \min_{x \in C} \frac{1}{T}\sum_{t=1}^{T} f_t(x) = \frac{\text{Regret}_T(\mathcal{A})}{T} = o(1).$$

In particular, in the offine setting where $f_1 = \cdots = f_T = f$, the statement is equivalent to

$$\frac{1}{T}\sum_{t=1}^{T} f(x_t^{\mathcal{A}}) - \min_{x \in C} f(x) = \frac{\text{Regret}_T(\mathcal{A})}{T} = o(1).$$

Hence, a sublinear regret means that the time averaged optimality gap goes to 0 as T increases.

3.2 Connection to Statistical Learning

In statistical learning, we are given a set of training data $z_1, \ldots, z_T \in \mathcal{Z}$ that are i.i.d. samples drawn from an unknown distribution $\mathcal{D}$. Based on the training data, we may choose a predictor $\bar{x} \in \mathcal{X}$. For some loss function $\ell : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$, the training error of $\bar{x}$ is given by

$$\bar{L}_T(\bar{x}) = \frac{1}{T}\sum_{t=1}^{T} \ell(\bar{x}, z_t),$$

while the generalization error of $\bar{x}$ is given by

$$L(\bar{x}) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(\bar{x}, z)].$$

The goal of statistical learning is to find a predictor $\bar{x}$ that minimizes the generalization error $L(\bar{x})$. Next, we argue that we may use an online algorithm to deduce a prediction with a small generalization error.

Algorithm 1 Online-to-Batch Reduction

Input: training data $z_1, \ldots, z_T$, online algorithm $\mathcal{A}$
for $t = 1, \ldots, T$ **do**
  take x_t from algorithm $\mathcal{A}$
  feed $\mathcal{A}$ with loss function $f_t(x) = \ell(x, z_t)$.
end for
sample τ from $\{1, \ldots, T\}$ uniformly at random.
return $\bar{x} = x_\tau$

Theorem 21.1. *Suppose that an online algorithm $\mathcal{A}$ guarantees*

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell(x_t, z_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} \ell(x, z_t)\right] \leq R(T).$$

Then

$$\mathbb{E}\left[L(\bar{x})\right] \leq L(x^*) + \frac{R(T)}{T}$$

where $x^* \in \mathit{argmin}_{x \in \mathcal{X}} L(x)$.

Proof. Since $\bar{x}$ is sampled from $\{x_1, \ldots, x_T\}$ uniformly at random, we have

$$\mathbb{E}\left[L(\bar{x})\right] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[L(x_t)\right] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}_{z \sim \mathcal{D}}\left[\ell(x_t, z)\right]\right] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\ell(x_t, z)\right].$$

Note that we have

$$\mathbb{E}\left[\ell(x_t, z)\right] = \mathbb{E}\left[\ell(x_t, z_t)\right]$$

because z_t is sampled from $\mathcal{D}$ after x_t is selected. Moreover, due to our assumption on the performance of $\mathcal{A}$, it follows that

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\ell(x_t, z_t)\right] &\leq \frac{1}{T}\mathbb{E}\left[\min_{x \in \mathcal{X}} \sum_{t=1}^{T} \ell(x, z_t)\right] + \frac{R(T)}{T} \\
&\leq \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \ell(x^*, z_t)\right] + \frac{R(T)}{T} \\
&= L(x^*) + \frac{R(T)}{T},
\end{aligned}$$

as required. □

4 Online Gradient Descent

There is a simple algorithm for online convex optimization that minimizes regret. In fact, a modification of gradient descent works for the online setting, and it is called online gradient descent.

The only distinction compared to the subgradient method for the offline setting is that we obtain a subgradient from the subdifferentials $\partial f_t(x_t)$ of functions f_t that are sequentially revealed. This simple algorithm does achieve an aymptotically optimal regret.

Algorithm 2 Online Gradient Descent (OGD)

Initialize $x_1 \in C$.
for $t = 1, \dots, T$ **do**
  Observe $f_t(x_t)$ and obtain $g_t \in \partial f_t(x_t)$.
  Obtain $x_{t+1} = \mathrm{proj}_C \{x_t - \eta_t g_t\}$ for a step size $\eta_t > 0$.
end for

Theorem 21.2. *Let $f_1, \dots, f_T$ be an arbitrary sequence of convex loss functions satisfying $\|g_t\|_2 \le L$ for any $g_t \in \partial f_t(x)$ for every $x \in \mathbb{R}^d$ and $t \ge 1$. Then online gradient descent given by Algorithm 2 with step sizes $\eta_t = R/(L\sqrt{t})$ where $R^2 = \sup_{x,y \in C} \|x - y\|_2^2$ satisfies*

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x \in C} \sum_{t=1}^{T} f_t(x) \le \frac{3}{2} LR\sqrt{T}.$$

Proof. The analysis of online gradient descent is quite similar to that of gradient descent. Let $x^* \in \mathrm{argmin}_{x \in C} \sum_{t=1}^{T} f_t(x)$. Note that

$$\begin{aligned}
\|x_{t+1} - x^*\|_2^2 &\le \|x_t - \eta_t g_t - x^*\|_2^2 \\
&= \|x_t - x^*\|_2^2 + \eta_t^2 \|g_t\|_2^2 - 2\eta_t g_t^\top (x_t - x^*) \\
&\le \|x_t - x^*\|_2^2 + \eta_t^2 L^2 - 2\eta_t (f_t(x_t) - f_t(x^*))
\end{aligned}$$

where the first inequality is due to the contraction property of the projection operator and the second inequality is due to the convexity of f_t. Then it follows that

$$f_t(x_t) - f_t(x^*) \le \frac{1}{2\eta_t} \left(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2 \right) + \frac{\eta_t}{2} L^2.$$

Adding up these inequalities for $t = 1, \dots, T$, we obtain

$$\begin{aligned}
\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x^*) &\le \sum_{t=1}^{T} \frac{1}{2\eta_t} \left(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2 \right) + \sum_{t=1}^{T} \frac{\eta_t}{2} L^2 \\
&\le \sum_{t=1}^{T} \|x_t - x^*\|_2^2 \left(\frac{1}{2\eta_t} - \frac{1}{2\eta_{t-1}} \right) + \frac{L^2}{2} \sum_{t=1}^{T} \eta_t \\
&\le \frac{R^2}{2} \sum_{t=1}^{T} \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) + \frac{L^2}{2} \sum_{t=1}^{T} \eta_t \\
&= \frac{R^2}{2} \cdot \frac{1}{\eta_T} + \frac{L^2}{2} \sum_{t=1}^{T} \frac{R}{L\sqrt{t}} \\
&\le \frac{3}{2} RL\sqrt{T}
\end{aligned}$$

where we set $1/\eta_0$ to be 0, the second inequality is because $\|x_{t+1} - x^*\|_2^2 \ge 0$, and the last inequality is because $\sum_{t=1}^{T} 1/\sqrt{t} \le 2\sqrt{T}$. □

Therefore, for Lipschitz continuous functions, OGD achieves the regret of $O(\sqrt{T})$. Can we do better than this?

Theorem 21.3. *Any algorithm for online convex optimization incurs $\Omega(LR\sqrt{T})$ regret in the worst case. The same statement holds even when the loss functions are i.i.d. with a fixed stationary distribution.*

For strongly convex and Lipschitz continuous functions, we can achieve a logarithmic regret!

Theorem 21.4. *Let $f_1, \ldots, f_T$ be an arbitrary sequence of convex loss functions satisfying $\|g_t\|_2 \leq L$ for any $g_t \in \partial f_t(x)$ for every $x \in \mathbb{R}^d$ and $t \geq 1$. Moreover, $f_1, \ldots, f_T$ are α-strongly convex with respect to the ℓ_2 norm. Then online gradient descent given by Algorithm 2 with step sizes $\eta_t = 1/(\alpha t)$ satisfies*

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x \in C} \sum_{t=1}^{T} f_t(x) \leq \frac{L^2}{2\alpha}(1 + \log T).$$