IE 539: Convex Optimization **KAIST, Fall 2023**
Lecture #14: Online and stochastic gradient descent algorithms October 25, 2023
Lecturer: Dabeen Lee

1 Outline

In this lecture, we study

- the online convex optimization (OCO) framework,
- online gradient descent for OCO,
- connections between OCO and stochastic optimization (SO),
- stochastic gradient descent.

2 Online convex optimization

In the last lecture, we discussed online advertisement selection, email spam filter, and multi-armed bandits. Remember that these problems involve a sequential decision-making that depends on interactions between decisions made by the decision-maker and data provided by the environment. As mentioned in the last lecture, 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.

In this section, we consider the extension of convex optimization to the online optimization setting. Namely, online convex optimization (OCO) is an online learning problem, that is to make a squence 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 modelling as well as convex optimization.

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.

2.1 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*.

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

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 $\mathrm{Regret}_T(\mathcal{A}) = o(T)$. What does this indicate? We look at the time averaged regret.

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

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

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

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

2.2 Online (sub)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.

Algorithm 1 Online gradient descent (OGD)

```
Initialize $x_1 \in C$.
for $t = 1, \ldots, T$ 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
```

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.

Theorem 14.1. *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 $t \geq 1$. Then online gradient descent given by Algorithm 1 with step sizes $\eta_t = R/(L\sqrt{t})$ where $R = \sup_{x,y\in C} \|x-y\|_2^2$ satisfies*

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

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

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

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

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

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

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

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

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

Theorem 14.2. *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 14.3. *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 1 with step sizes $\eta_t = 1/(\alpha t)$ satisfies*

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

2.3 Stochastic vs adversarial

Where do $f_1, \ldots, f_T$ come from?

- (Stochastic i.i.d.) There is a distribution of functions, and at each time step, a function is sampled from the distribution independently from the history. Here, $f_1, \ldots, f_T$ are independent and identically distributed (i.i.d.).
- (Markov chain) $f_1, \ldots, f_T$ are sampled from a Markov chain.
- (General stochastic process) $f_1, \ldots, f_T$ form a general stochastic process.
- (Adaptive adversary) There is an adaptive adversary or an environment that can observe the history of decisions, based on which it chooses the next loss function.

Basically, the problem is that we make decisions to reduce our loss, but at the same time, the environment can choose loss functions to increase our loss. With this regard, the stochastic i.i.d. setting and the adversarial setting are different. We can imagine that an adptive adversary can make our loss worse than the non-adaptive stochastic sampling of loss functions.

2.4 Stochastic optimization

Stochastic optimization (SO) is an optimization problem of the following form.

$$\underset{x\in C}{\text{minimize}} \quad \mathbb{E}_{\xi\sim\mathbb{P}}\left[h(x,\xi)\right]$$

where

- ξ is a random parameter vector whose underlying distribution is given by $\mathbb{P}$,
- $h(x,\xi)$ is convex with respect to x for any fixed ξ,
- C is the feasible set for the decision vector x.

Then

$$f(x) = \mathbb{E}_{\xi\sim\mathbb{P}}\left[h(x,\xi)\right]$$

is convex. For example, for the linear regression problem, we consider

$$h(\beta,(x,y)) = \frac{1}{2}(y-\beta^\top x)^2,$$

and

$$\text{minimize} \quad \mathbb{E}_{(x,y)\sim\mathbb{P}}\left[h(\beta,(x,y))\right] \quad = \quad \text{minimize} \quad \mathbb{E}_{(x,y)\sim\mathbb{P}}\left[\frac{1}{2}(y-\beta^\top x)^2\right]$$

where x is the feature vector, y is the response variable, and (x,y) follows distribution $\mathbb{P}$.

We can solve the stochastic optimization problem based on online convex optimization. At each iteration t, we sample a random parameter vector ξ_t from $\mathbb{P}$, based on which we update our decision vector. To be more precise, we start with a decision vector $x_1 \in C$. Then we obtain a random vector ξ_1, and we adjust our decision vector to obtain a new decision vector x_2. We repeat this procedure for T time steps. We can relate this process to the online convex optimization framework. For $t = 1, \ldots, T$, we define f_t as

$$f_t(x) = h(x,\xi_t).$$

We decide x_t, after which we observe random vector ξ_t. Hence, we can look at

$$\sum_{t=1}^T f_t(x_t) - \min_{x\in C}\sum_{t=1}^T f_t(x) = \sum_{t=1}^T h(x_t,\xi_t) - \min_{x\in C}\sum_{t=1}^T h(x,\xi_t),$$

which is the regret of the corresponding online convex optimization problem. How does it relate to solving the stochastic optimization problem? For stochastic optimization, we consider the average of $x_1,\ldots,x_T$ as a candidate solution and compute the optimality gap given by

$$\mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[f\left(\frac{1}{T}\sum_{t=1}^T x_t\right)\right] - f(x^*)$$

where the expectation is taken over the randomness in choosing $x_2,\ldots,x_T$ and $x^*\in\operatorname{argmin}_{x\in C} f(x)$.

Theorem 14.4. *The optimality gap for SO and the regret for OCO satisfy the following relation.*

$$\mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[f\left(\frac{1}{T}\sum_{t=1}^T x_t\right)\right] - f(x^*) \le \frac{1}{T}\mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[\sum_{t=1}^T f_t(x_t) - \min_{x\in C}\sum_{t=1}^T f_t(x)\right].$$

Proof. First we deduce that

$$\begin{aligned}\mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[f\left(\frac{1}{T}\sum_{t=1}^T x_t\right)\right] - f(x^*) &\le \mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[\frac{1}{T}\sum_{t=1}^T f\left(x_t\right)\right] - f(x^*)\\ &= \frac{1}{T}\sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[f\left(x_t\right)\right] - \frac{1}{T}\sum_{t=1}^T f(x^*)\\ &= \frac{1}{T}\sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_{t-1}\sim\mathbb{P}}\left[f\left(x_t\right)\right] - \frac{1}{T}\sum_{t=1}^T f(x^*).\end{aligned}$$

Note that

$$\begin{aligned}&\sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_{t-1}\sim\mathbb{P}}\left[f\left(x_t\right)\right] - \sum_{t=1}^T f(x^*)\\ &= \sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_{t-1}\sim\mathbb{P}}\left[\mathbb{E}_{\xi_t\sim\mathbb{P}}\left[h(x_t,\xi_t)\mid \xi_1,\ldots,\xi_{t-1}\right]\right] - \sum_{t=1}^T \mathbb{E}_{\xi_t\sim\mathbb{P}}\left[h(x^*,\xi_t)\right]\\ &= \sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_t\sim\mathbb{P}}\left[h(x_t,\xi_t)\right] - \sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_t\sim\mathbb{P}}\left[h(x^*,\xi_t)\right]\\ &= \sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[h(x_t,\xi_t)\right] - \sum_{t=1}^T \mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[h(x^*,\xi_t)\right]\\ &= \mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[\sum_{t=1}^T h(x_t,\xi_t) - \sum_{t=1}^T h(x^*,\xi_t)\right]\\ &= \mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[\sum_{t=1}^T f_t(x_t) - \sum_{t=1}^T f_t(x^*)\right]\\ &\le \mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[\sum_{t=1}^T f_t(x_t) - \min_{x\in C}\sum_{t=1}^T f_t(x)\right].\end{aligned}$$

Therefore, we obtain

$$\mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[f\left(\frac{1}{T}\sum_{t=1}^{T}x_t\right)\right]-f(x^*)\leq\frac{1}{T}\mathbb{E}_{\xi_1,\ldots,\xi_T\sim\mathbb{P}}\left[\sum_{t=1}^{T}f_t(x_t)-\min_{x\in C}\sum_{t=1}^{T}f_t(x)\right],$$

which provides an upper bound on the optimality gap for stochastic optimization. □

This in turn implies that by designing an online algorithm that minimizes the regret term for the online convex optimization problem, we can solve the stochastic optimization problem. The following is the application of online gradient descent to our stochastic optimization setting.

Algorithm 2 Online gradient descent for stochastic optimization

```
Initialize $x_1 \in C$.
for $t = 1, \ldots, T$ do
    Obtain a random vector $\xi_t \sim \mathbb{P}$ and a subgradient $g(x_t, \xi_t) \in \partial h(x_t, \xi_t)$.
    Obtain $x_{t+1} = \mathrm{Proj}_C\{x_t - \eta_t g(x_t, \xi_t)\}$ for a step size $\eta_t > 0$.
end for
```

In fact, Algorithm 2 is the so-called stochastic gradient descent method. In particular, it is well-known that

$$\mathbb{E}_{\xi_t\sim\mathbb{P}}\left[g(x_t,\xi_t)\right]\in\partial f(x_t),$$

which means that $g(x_t,\xi_t)$ is an unbiased estimator of a subgradient of f at x_t. This is what we need for the convergence of stochastic gradient descent!

3 Stochastic gradient descent

Although stochastic gradient descent (SGD) on its own is a very important subject of study in optimization and machine learning, we present it as an application of online gradient descent. This section will be a gentle introduction to SGD.

Let us get back to the offline convex optimization stated as

$$\min_{x\in C} f(x).$$

If we have an access to its gradient or one of its subgradients, then we can apply gradient descent or the subgradient method. However, depending on situations, it may not be realistic to assume that we have an oracle that provides exact gradients. For example, we have just considered the stochastic optimization setting where f is given by $f(x)=\mathbb{E}_{\xi\sim\mathbb{P}}\left[h(x,\xi)\right]$, the expectation of a random function. Here, $\nabla f(x)=\mathbb{E}_{\xi\sim\mathbb{P}}\left[\nabla h(x,\xi)\right]$, to compute which we need to know the distribution $\mathbb{P}$ in general. Instead of computing the expectation exactly, what we did was to obtain a sample ξ_t so that we may use $\nabla h(x,\xi_t)$ for each iteration t. Here $\nabla h(x,\xi_t)$ is an unbiased estimator of $\nabla f(x)$.

Another example is the mean squared error minimization problem for regression.

$$\min_{\beta}\quad f(\beta)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i-\beta^\top x_i)^2$$

where $(x_1,y_1),\ldots,(x_n,y_n)$ are the given data. In fact, this setting is also a stochastic optimization problem as we can define $\mathbb{P}$ as the empirical distribution over the n samples. To be more specific,

$$\mathbb{P}\left((x,y)=(x_i,y_i)\right)=\frac{1}{n}.$$

Then the gradient of f at β is given by

$$\nabla f(\beta) = \mathbb{E}_{(x,y)\sim\mathbb{P}}[\nabla h(\beta, (x,y))] = -\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)x_i.$$

In this example, we know the precise description of the underlying distribution, from which we can compute the exact gradient. Then, what is the problem? Here, to compute the gradient, we have to go through all data points $(x_1, y_1), \ldots, (x_n, y_n)$, which may not be practical especially when the number of data is large. For this scenario, a strategy is to obtain an estimation of the gradient. We sample a data (x_r, y_r) from the data set uniformly at random and obtain

$$g_r = -(y_r - \beta^\top x_r)x_r.$$

Here r is a random variable following the uniform distribution over $\{1, \ldots, n\}$. Note that

$$\mathbb{E}[g_r] = \sum_{i=1}^{n}\mathbb{P}(r = i)\cdot g_i = \sum_{i=1}^{n}\frac{1}{n}\cdot g_i = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)x_i = \nabla f(\beta).$$

Hence, g_r is an unbiased estimator of g_r. What we do next is to use g_r to replace $\nabla f(\beta)$ when running gradient descent. More generally, let $\tilde{g}_x$ be an unbiased estimator of the gradient of f at x or the subgradient for f at x.

Algorithm 3 Stochastic gradient descent (SGD)

```
Initialize $x_1 \in C$.
for $t = 1, \ldots, T$ do
    Obtain an estimator $\hat{g}_{x_t}$ of some $g \in \partial f(x_t)$.
    Update $x_{t+1} = \mathrm{Proj}_C\{x_t - \eta_t \hat{g}_{x_t}\}$ for a step size $\eta_t > 0$.
end for
Return $(1/T)\sum_{t=1}^{T} x_t$.
```

Assume that $\tilde{g}_x$ satisfies

$$\mathbb{E}[\hat{g}_x] = g \text{ for some } g \in \partial f(x), \quad \mathbb{E}\left[\|\hat{g}_x\|^2\right] \leq L^2.$$

Under this assumption, let us analyze the performance of stochastic gradient descent given by Algorithm 3.

Theorem 14.5. *Algorithm 3 with step sizes $\eta_t = R/(L\sqrt{t})$ satisfies*

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right)\right] - f(x^*) \leq \frac{3LR}{2\sqrt{T}}$$

where the expectation is taken over the randomness in gradient estimation and $x^* \in \mathrm{argmin}_{x\in C} f(x)$.

Proof. Suppose that $\mathbb{E}[\tilde{g}_{x_t}] = g_t \in \partial f(x_t)$ for $t \geq 1$. First, let us observe the following.

$$\begin{aligned}
\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right)\right] - f(x^*) &\leq \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} f(x_t)\right] - f(x^*) \\
&= \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}(f(x_t) - f(x^*))\right] \\
&\leq \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} g_t^\top (x_t - x^*)\right] \\
&= \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \tilde{g}_{x_t}^\top (x_t - x^*)\right]
\end{aligned}$$

where the inequalities are due to the convexity of f. Now let us consider functions $f_1, \ldots, f_T$ given by

$$f_t(x) = \tilde{g}_{x_t}^\top x.$$

Then

$$\begin{aligned}
\sum_{t=1}^{T} \tilde{g}_{x_t}^\top (x_t - x^*) &= \sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(x^*) \\
&\leq \sum_{t=1}^{T} f_t(x_t) - \min_{x \in C} \sum_{t=1}^{T} f_t(x) \\
&\leq \frac{3}{2} LR\sqrt{T}
\end{aligned}$$

where the last inequality is from the convergence result of online gradient descent. Note that this upper bound holds regardless of any realization of $\tilde{g}_{x_t}$'s. Therefore, the result follows. $\square$

References

[DSSSC08] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 272–279, New York, NY, USA, 2008. Association for Computing Machinery.

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[Haz16] Elad Hazan. Introduction to online convex optimization. *Found. Trends Optim.*, 2(3–4):157–325, aug 2016.

[JS10] Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 471–478, Madison, WI, USA, 2010. Omnipress.