IE 539: Convex Optimization **KAIST, Fall 2022**
Lecture #13: Accelerated method and Online convex optimization October 13, 2022
Lecturer: Dabeen Lee

1 Outline

In this lecture, we study

- Projected gradient descent,
- Conditional gradient method,
- Lower bounds on the iteration complexity of gradient methods.

2 Accelerated gradient method

From the last lecture, we learned that for smooth functions, there is some gap between the convergence rate of gradient descent and the oracle lower bound. Then the question we asked was as to whether we can find an algorithm that achieves a better convergence rate or improve the lower bound. The answer to the question is that there is a better algorithm, which closes the gap, thereby achieving the optimal asymptotic convergence rate. The algorithm is due to Nesterov [Nes83, Nes04], and it is referred to as Nesterov's accelerated gradient descent. Let us describe the algorithm and explain how it achieves a better convergence rate.

The main idea behind Nesterov's acceleration is to use "momentum", so the algorithm is often called gradient descent with momentum. Recall that gradient descent for a β-smooth function follows the update rule of

$$x_{t+1} = x_t - \frac{1}{\beta}\nabla f(x_t)$$

from a given point x_t. The idea of momentum is to incorporate the direction $x_t - x_{t-1}$ that we took when moving from x_{t-1} to x_t to obtain the next iterate x_{t+1}. Then x_{t+1} is determined by not only the previous iterate x_t but also x_{t-1} which is the one before x_t. Figure 13.1 illustrates how the idea of momentum applies. Instead of applying the gradient descent update to x_t, we move a

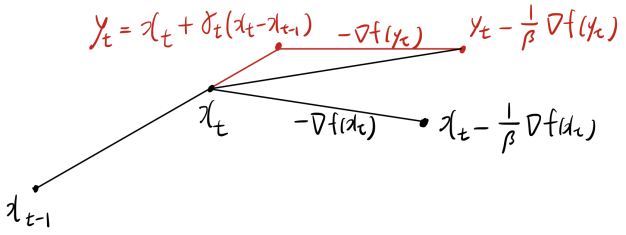
$y_t = x_t + \gamma_t(x_t - x_{t-1})$
$-\nabla f(y_t)$
$y_t - \frac{1}{\beta}\nabla f(y_t)$
x_t
$-\nabla f(x_t)$
$x_t - \frac{1}{\beta}\nabla f(x_t)$
x_{t-1}

Figure 13.1: Illustration of gradient descent with momentum

bit further from x_t along the momentum direction that we took from x_{t-1} to x_t. Let $\gamma_t > 0$ be a weight, and

$$y_t = x_t + \gamma_t(x_t - x_{t-1}).$$

Then we apply the gradient descent update on y_t to obtain the next point x_{t+1}, as follows.

$$x_{t+1} = y_t - \frac{1}{\beta}\nabla f(y_t).$$

This is basically the outline of Nesterov's accelerated gradient descent.

Algorithm 1 Nesterov's accelerated gradient descent

```
Initialize $x_1 \in \text{dom}(f)$.
Set $x_0 = x_1$.
for $t = 1, \ldots, T$ do
    $y_t = x_t + \gamma_t(x_t - x_{t-1})$ for some $\gamma_t > 0$.
    $x_{t+1} = y_t - \frac{1}{\beta}\nabla f(y_t)$.
end for
Return $x_{T+1}$.
```

The following shows a convergence result of the accelerated gradient descent method for smooth functions.

Theorem 13.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a β-smooth convex function with respect to the ℓ_2 norm. We set γ_t by the following procedure.*

$$\lambda_0 = 0, \quad \lambda_t = \frac{1 + \sqrt{1 + 4\lambda_{t-1}^2}}{2}, \quad \gamma_t = \frac{\lambda_t - 1}{\lambda_{t+1}}.$$

Then

$$f(x_T) - f(x^*) \le \frac{2\beta\|x_1 - x^*\|_2^2}{T^2}$$

where x^* is an optimal solution to $\min_{x \in \mathbb{R}^d} f(x)$.

Hence, the convergence rate is $O(1/T^2)$, which matches the oracle lower bound. The number of required iterations to bound the error by ϵ is $O(1/\sqrt{\epsilon})$. The next result is for functions that are both smooth and strongly convex.

Theorem 13.2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function that is β-smooth and α-strongly convex with respect to the ℓ_2 norm. We set*

$$\gamma_t = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$$

where $\kappa = \beta/\alpha$. Then

$$f(x_T) - f(x^*) \le \frac{\alpha + \beta}{2} \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{(T-1)/2} \|x_1 - x^*\|_2^2.$$

where x^* is an optimal solution to $\min_{x \in \mathbb{R}^d} f(x)$.

3 Online convex optimization

We have so far discussed formulations and algorithms for convex optimization. In this section, we consider a different yet closely related setting. 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 contents of this section are based on the text of Hazan [Haz16].

Before we discuss some specific examples, let us provide a concrete problem formulation. 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.

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 Examples

Let us provide some applications of the framework.

- (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 tunrs 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 the user demands.

3.2 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.).
- (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. In contrast, the stochastic i.i.d. setting is non-adaptive.

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.

3.3 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.

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

In the previous subsection, we discussed the role of adaptive adversary in terms of choosing the loss functions $f_1, \ldots, f_T$. Basically, an adaptive adversary may select the worst loss functions that deteriorate the cumulative loss of algorithm $\mathcal{A}$. Motivated by this, we defined the regret of $\mathcal{A}$ as the worst-case loss gap as follows.

$$\mathrm{Regret}_T(\mathcal{A}) = \sup_{f_1,\ldots,f_T} \left\{ \sum_{t=1}^{T} f_t(x_t^{\mathcal{A}}) - \min_{x \in C} \sum_{t=1}^{T} f_t(x) \right\}.$$

We focus on developing algorithms that minimize the regret, for which we can analyze the notion of 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 optimality gap.

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

Hence, a sublinear regret means that the time averaged optimality gap goes to 0 as T increases. In particular, in the offine setting where $f_1 = \cdots = f_T = f$, the statement is equivalent to

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

which corresponds to a convergence of gradient descent.

3.4 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 2 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 (sub)gradient 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 13.3. *Suppose that $\|g_t\|_2 \le L$ 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 $R = \sup_{x,y\in C}\|x-y\|_2^2$ 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 $x^* \in \operatorname{argmin}_{x\in C}\sum_{t=1}^T f_t(x)$. 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 13.4. *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 generated from a fixed stationary distribution (the stochatic i.i.d. setting).*

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

Theorem 13.5. *Suppose that $\|g_t\|_2 \le L$ for any $g_t \in \partial f_t(x)$ for every $x \in \mathbb{R}^d$ and $t \ge 1$. Moreover, $f_1,\dots,f_T$ 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*

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

3.5 Application: 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, consider the mean squared error minimization problem for regression.

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

where $(x_1, y_1), \ldots, (x_n, y_n)$ are the given data. Then the gradient of f at β is given by

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

What is the problem? Here, to compute the gradient, we have to go through all data points $(x_r, y_r), \ldots, (x_r, y_r)$, 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_i, y_i) from the data set uniformly at random and obtain

$$g_r = -2(y_r - \beta^\top x_r)x_r.$$

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

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

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

Theorem 13.6. *Algorithm 3 with step sizes $\eta_t = R/(L\sqrt{t})$ satisfies*

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

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 $t \ge 1$. First, let us observe the following.

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

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

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

where the last inequality is from Theorem 13.3. Note that this upper bound holds regardless of any realization of $\tilde{g}_{x_t}$'s. Therefore, the result follows. □

References

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

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983. 2

[Nes04] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course.* Kluwer Academic Publishers, Norwell, 2004. 2