IE 539: Convex Optimization **KAIST, Fall 2023**
Lecture #12: Projected gradient methods, Acceleration: gradient descent with momentum, Oracle complexity lower bounds October 11, 2023
Lecturer: Dabeen Lee

1 Outline

In this lecture, we study

- Projected gradient descent for constrained minimization.
- Lower bounds on the iteration complexity of a first-order method,
- Accelerated method: gradient descent with momentum,

2 Projected gradient descent

So far, we considered gradient descent for unconstrained convex minimization under various settings. Gradient descent proceeds with the update rule

$$x_{t+1} = x_t - \eta_t \nabla f(x_t).$$

If f is not differentiable, we may take a subgradient $g \in \partial f(x_t)$ at x_t instead of the gradient.

For the constrained case, however, the update rule does not necessarily generate a feasible solution. A natural fix for this is that we take the projection of the point $x_t - \eta_t \nabla f(x_t)$ onto the feasible set

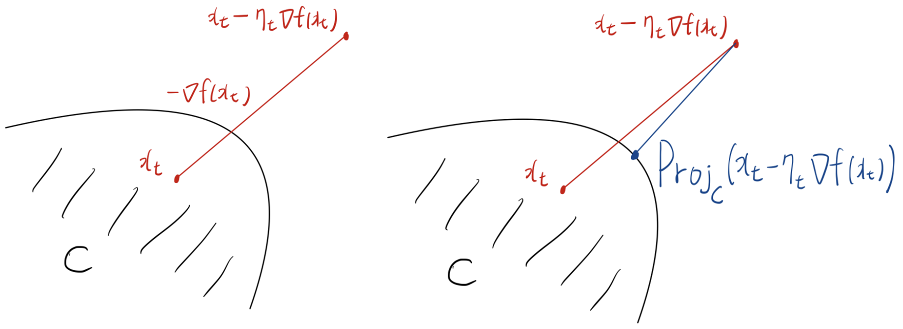

$x_t - \eta_t \nabla f(x_t)$
$-\nabla f(x_t)$
x_t
C
$x_t - \eta_t \nabla f(x_t)$
x_t
$\mathrm{Proj}_C(x_t - \eta_t \nabla f(x_t))$
C

Figure 12.1: Infeasible point after a gradient descent update and projection

C. What we have just discussed is basically the projected gradient descent method! It is basically gradient descent with projection. To formalize, let us give a pseudo-code of the projected gradient descent method. In Algorithm 1, we use the operator $\mathrm{Proj}_C(\cdot)$, which is formally defined as

$$\mathrm{Proj}_C(z) = \operatorname*{argmin}_{x \in C} \frac{1}{2}\|x - z\|_2^2 \quad \text{for } z \in \mathbb{R}^d.$$

Then it is straightforward that

$$\mathrm{Proj}_C(z) = \operatorname*{argmin}_{x \in C} \|x - z\|_2,$$

Algorithm 1 Projected gradient descent method

```
Initialize $x_1 \in C$.
for $t = 1, \ldots, T$ do
    $x_{t+1} = \mathrm{Proj}_C \{x_t - \eta_t \nabla f(x_t)\}$ for a step size $\eta_t > 0$.
end for
Return $x_{T+1}$.
```

and in words, $\mathrm{Proj}_C(z)$ is a point C that is closest to point z with respect to the ℓ_2 norm distance. Although we have discussed the following lemma in a previous lecture, we include it again to make this note self-contained.

Lemma 12.1. *Let $x \in C$ and $z \in \mathbb{R}^d$. Then*

$$(\mathit{Proj}_C(z) - z)^\top (\mathit{Proj}_C(z) - x) \leq 0 \quad \text{for all } x \in C.$$

Proof. We can apply the optimality condition to the definition $\mathrm{Proj}_C(z) = \mathrm{argmin}_{x\in C} \frac{1}{2}\|x - z\|_2^2$ for $z \in \mathbb{R}^d$. The gradient of $\frac{1}{2}\|x - z\|_2^2$ at $x = \mathrm{Proj}_C(z)$ is $(\mathrm{Proj}_C(z) - z)$. Then the statement is precisely the optimality condition for $\mathrm{Proj}_C(z)$. □

By definition, x_{t+1} is the point in C that is closest to $x_t - \eta_t \nabla f(x_t)$ with respect to the ℓ_2 distance. Moreover, we have another interpretation of the update rule based on the following.

$$\begin{aligned} x_{t+1} &= \underset{x\in C}{\mathrm{argmin}} \left\{ \frac{1}{2}\|x - x_t + \mu_t \nabla f(x_t)\|_2^2 \right\} \\ &= \underset{x\in C}{\mathrm{argmin}} \left\{ f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{1}{2\eta_t}\|x - x_t\|_2^2 \right\}, \end{aligned}$$

which means that x_{t+1} is the solution in C minimizing the quadratic approximation of f at x_t.

Hereinafter, we introduce notations y_{t+1} to denote $x_t - \eta_t \nabla f(x_t)$ for simpler presentations. Then the update rule can be written as

$$\begin{aligned} y_{t+1} &= x_t - \eta_t \nabla f(x_t), \\ x_{t+1} &= \mathrm{Proj}_C(y_{t+1}) \end{aligned}$$

for $t = 1, \ldots, T$. The analysis of projected gradient descent is quite similar to that of gradient descent for unconstrained minimization. The following is useful to make the analysis for gradient descent go through for the case of projected gradient descent.

Lemma 12.2. *For any t, we have*

$$\|x_{t+1} - x^*\|_2 \leq \|y_{t+1} - x^*\|_2$$

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

Proof. We use Lemma 12.1 and the fact that $x_{t+1} = \mathrm{Proj}_C(y_{t+1})$. By Lemma 12.1,

$$(x_{t+1} - y_{t+1})^\top (x_{t+1} - x^*) \leq 0.$$

Since $x_{t+1} - y_{t+1} = x_{t+1} - x^* + x^* - y_{t+1}$, the inequality implies that

$$\|x_{t+1} - x^*\|_2^2 \leq (y_{t+1} - x^*)^\top (x_{t+1} - x^*) \leq \|y_{t+1} - x^*\|_2 \|x_{t+1} - x^*\|_2$$

where the last inequality is due to the Cauchy-Schwarz inequality. Dividing each side by $\|x_{t+1} - x^*\|_2$, we obtain the result. □

By Lemma 12.2, we deduce that

$$\begin{aligned}\|x_{t+1}-x^*\|_2^2 &\leq \|y_{t+1}-x^*\|_2\\ &= \|x_t-x^*\|_2^2 - 2\eta_t\nabla f(x_t)^\top(x_t-x^*) + \eta_t^2\nabla f(x_t)^2\\ &\leq \|x_t-x^*\|_2^2 - 2\eta_t(f(x_t)-f(x^*)) + \eta_t^2\nabla f(x_t)^2,\end{aligned}$$

which appears in the convergence analysis of gradient descent for Lipschitz continuous functions. Note that the only difference from the unconstrained case is the first inequality, which used to be an equality for the unconstrained case where $y_{t+1} = x_{t+1}$. Based on this, we recover the same convergence theorem for projected gradient descent for the case of Lipschitz continuous functions. In fact, we can work over the projected subgradient method, which is as the name suggests the subgradient method with projection for the constrained minimization.

Algorithm 2 Projected subgradient method

```
Initialize $x_1 \in C$.
for $t = 1, \ldots, T$ do
    Obtain a subgradient $g_t \in \partial f(x_t)$.
    $x_{t+1} = \mathrm{Proj}_C\{x_t - \eta_t g_t\}$ for a step size $\eta_t > 0$.
end for
Return $(\sum_{t=1}^T \eta_t)^{-1}\sum_{t=1}^T \eta_t x_t$.
```

The following theorem shows the convergence of the projected subgradient method for functions that have bounded subgradients.

Theorem 12.3. *Let $f:\mathbb{R}^d \to \mathbb{R}$ be a convex function such that $\|g\|_2 \leq L$ for any $g \in \partial f(x)$ for every $x \in \mathbb{R}^d$. Let $\{x_t : t = 1,\ldots,T\}$ be the sequence of iterates generated by the projected subgradient method with step size $\eta_t = \|x_1 - x^*\|_2/L\sqrt{T}$ for each t. Then*

$$f\left(\frac{1}{T}\sum_{t=1}^T x_t\right) - f(x^*) \leq \frac{L\|x_1-x^*\|_2}{\sqrt{T}}$$

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

Moreover, we also recover the same "asymptotic" convergence rate for strongly convex, smooth, and strongly convex & smooth functions. In particular,

Theorem 12.4. *Let $f:\mathbb{R}^d \to \mathbb{R}$ be a β-smooth convex function, and let $\{x_t : t = 1,\ldots,T\}$ be the sequence of iterates generated by gradient descent with step size $\eta_t = 1/\beta$ for each t. Then*

$$f(x_T) - f(x^*) \leq \frac{3\beta\|x_1-x^*\|_2^2 + f(x_1) - f(x^*)}{T}$$

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

3 Lower bounds on the iteration complexity of gradient methods

We discussed the convergence rates of gradient descent and the subgradient method. In particular, for Lipschitz continuous functions, we know that the subgradient method guarantees the convergence rate of $O(1/\sqrt{T})$ and requires $O(1/\epsilon^2)$ iterations to achieve the error bounded by ϵ. For smooth convex functions, gradient descent achieves $O(1/T)$ convergence rate, and the number of required iterations to bound the error by ϵ is $O(1/\epsilon)$. For functions that are both smooth and strongly convex, the convergence rate of gradient descent is $O(\gamma^T)$ for some $0 < \gamma < 1$, and the number of required iterations is $O(\log(1/\epsilon))$ to achieve an error of ϵ.

A natural question is as to whether we can find an algorithm that achieves a better convergence rate. Regarding this question, we conceptualize the (first-order) oracle complexity of an algorithm. A first-order oracle for convex minimization $\min_{x\in C} f(x)$ takes a point x in C as an input and

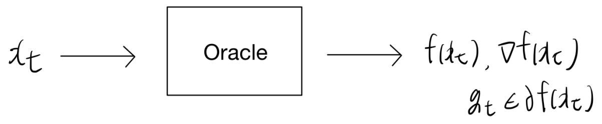
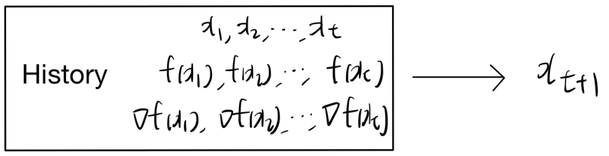


Figure 12.2: Oracle that returns the function value and the first-order information

returns its function value $f(x)$ as well as the first-order information, i.e., the gradient $\nabla f(x)$ or a subgradient $g_t \in \partial f(x)$. Then the oracle complexity of an oracle-based algorithm counts the number of oracle calls to terminiate. An oracle-based algorithm can be illustrated as follows. Basically, it



History
$x_1, x_2, \cdots, x_t$
$f(x_1), f(x_2), \cdots, f(x_t)$
$\nabla f(x_1), \nabla f(x_2), \cdots, \nabla f(x_t)$
x_{t+1}

Figure 12.3: Illustration of an oracle-based algorithm

picks a new solution based on the history of past iterates and their first-order information.

We present some lower bound results on the oracle complexity given by Nemirovski and Yudin in 1983 [NY83] (see also Nesterov [Nes03] and Bubeck [Bub15]). We make the assumption that $x_1 = 0$ and x_{t+1} belongs to the span of $g_1, \ldots, g_t$ where $g_s \in \partial f(x_s)$.

Theorem 12.5 (See [Bub15]). *There exists a convex and L-Lipschitz continuous function $f : \mathbb{R}^d \to \mathbb{R}$ for some $L > 0$ such that iterates $x_1, \ldots, x_t$ with $t \leq d$ generated by any oracle-based algorithm satisfies the following:*

$$\min_{1\leq s\leq t} f(x_s) - \min_{x\in B_2(R)} f(x) \geq \frac{RL}{2(1+\sqrt{t})}$$

where $B_2(R) = \{x \in \mathbb{R}^d : \|x\|_2 \leq R\}$ and $R > 0$.

Theorem 12.6 (See [Bub15]). *There exists a convex and β-smooth fuction $f : \mathbb{R}^d \to \mathbb{R}$ with respect to the ℓ_2-norm for some $\beta > 0$ such that iterates $x_1, \ldots, x_t$ with $t \leq (d-1)/2$ generated by any oracle-based algorithm satisfies the following:*

$$\min_{1\leq s\leq t} f(x_s) - \min_{x\in\mathbb{R}^d} f(x) \geq \frac{3\beta\|x_1 - x^*\|_2^2}{32(t+1)^2}.$$

Theorem 12.7 (See [Bub15]). *There exists a β-smooth and α-strongly convex fuction $f : \mathbb{R}^d \to \mathbb{R}$ with respect to the ℓ_2-norm for some $\beta \geq \alpha > 0$ such that x_t with $t \geq 1$ generated by any oracle-based algorithm satisfies the following:*

$$f(x_t) - \min_{x\in\mathbb{R}^d} f(x) \geq \frac{\alpha}{2}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(t-1)} \|x_1 - x^*\|_2^2.$$

4 Accelerated gradient method

We just argued in the previous section that for smooth functions, there is some gap between the convergence rate of gradient descent and the oracle lower bound. Can we find an algorithm that achieves a better convergence rate or improve the lower bound? The answer to the question is that there is indeed 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 12.4 illustrates how the idea of momentum applies. Instead of applying the gradient descent update to x_t, we move a

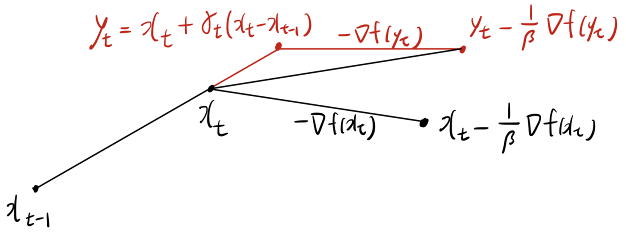

$y_t = x_t + \gamma_t(x_t - x_{t-1})$
$-\nabla f(y_t)$
$y_t - \frac{1}{\beta}\nabla f(y_t)$
x_t
$-\nabla f(x_t)$
$x_t - \frac{1}{\beta}\nabla f(x_t)$
x_{t-1}

Figure 12.4: 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.

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

Algorithm 3 summarizes Nesterov's accelerated gradient descent that we just explained. The fol-

Algorithm 3 Nesterov's accelerated gradient descent

```
Initialize $x_1 \in \mathrm{dom}(f)$.
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 $x_{T+1}$.
```

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

Theorem 12.8. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a β-smooth convex function in the ℓ_2 norm. We set γ_t by the following procedure.*

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

Then

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

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

For example, we may set

$$\lambda_t = \frac{t+2}{2}, \quad t \geq 0.$$

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

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

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

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

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

References

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3–4):231–357, 2015. 3, 12.5, 12.6, 12.7

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. 5

[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. 4

[Nes03] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003. 3

[Nes04] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course.* Kluwer Academic Publishers, Norwell, 2004. 4

[NY83] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983. 3