IE 539: Convex Optimization **KAIST, Fall 2024**
Lecture #12: Lower bounds, Accelerated gradient descent, Frank-Wolfe October 28, 2024
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 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

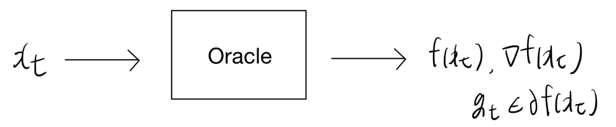

Figure 12.1: Oracle that returns the function value and the first-order information

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

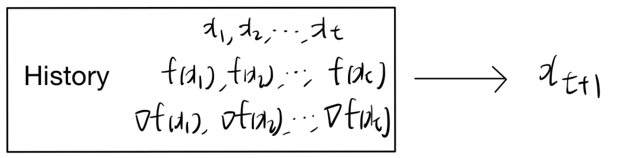
History
$x_1, x_2, \cdots, x_t$
$f(x_1), f(x_2), \cdots, f(x_t)$
$\nabla f(x_1), \nabla f(x_2), \cdots, \nabla f(x_t)$
x_{t+1}

Figure 12.2: 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 12.1 (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 $t \le d$ generated by any oracle-based algorithm satisfies the following:*

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

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

Theorem 12.2 (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 $t \le (d-1)/2$ generated by any oracle-based algorithm satisfies the following:*

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

Theorem 12.3 (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 $\beta \ge \alpha > 0$ such that x_t with $t \ge 1$ generated by any oracle-based algorithm satisfies the following:*

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

3 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 12.3 illustrates how the idea of momentum applies. Instead of applying the gradient descent update to x_t, we move a 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.

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

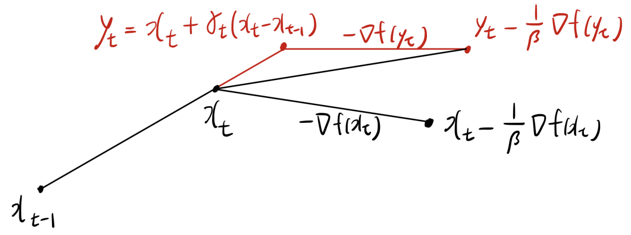
$y_t = x_t + \gamma_t(x_t - x_{t-1})$
$-\nabla f(y_t)$
$y_t - \frac{1}{\beta}\nabla f(y_t)$
x_t
$-\nabla f(x_t)$
$x_t - \frac{1}{\beta}\nabla f(x_t)$
x_{t-1}

Figure 12.3: Illustration of gradient descent with momentum

Algorithm 1 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 $x_{T+1}$.
```

Algorithm 1 summarizes Nesterov's accelerated gradient descent that we just explained. The following shows a convergence result of the accelerated gradient descent method for smooth functions.

Theorem 12.4. *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.*

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

Then

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

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

For example, we may set

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

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 12.5. *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

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

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

4 Projection-free method

We saw that projected gradient descent minimizes a smooth function with a convergence rate of $O(1/T)$. There are some issues.

1. The projection step onto the feasible set C can be expensive.
2. We have used the ℓ_2 norm to define smoothness.

Each projection step essentially amounts to solving an optimization problem, which can be difficult depending on the structure of C. Even for the case when C is a polyhedron, the projection onto C can an expensive procedure. The second point is that in our analysis of gradient descent for smooth functions, there are parts that do need smoothness with respect to the ℓ_2 norm. It is often the case that smoothness in the ℓ_2 norm is implied by smoothness in another norm, e.g., the ℓ_1 norm.

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \sqrt{d}\|\nabla f(x) - \nabla f(y)\|_\infty \leq \sqrt{d}\beta\|x-y\|_1 \leq d\beta\|x-y\|_2.$$

The implication of this inequality is the following. Even if a function is smooth in the ℓ_1 norm with a tiny smoothness parameter β, the smoothness parameter with respect to the ℓ_2 norm can blow up by a factor of dimension d, in which case we lose the desired dimension-free property.

4.1 Constrained optimization formulation of LASSO

We studied the folllowing formulation of LASSO given n data points n data points $(x_1, y_1), \ldots, (x_n, y_n)$.

$$\min_\beta \quad \frac{1}{n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

where

- $y = (y_1, \ldots, y_n)^\top$ and the rows of X are $x_1^\top, \ldots, x_n^\top$,
- the quadratic term
$$\frac{1}{n}\sum_{i=1}^n (y_i - \beta^\top x_i)^2 \quad = \quad \frac{1}{n}\|y - X\beta\|_2^2$$
is the mean squared error of regressor β.

Recall that the formulation is the *Lagrangian form* of LASSO. The constrained form is given by

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n}\|y - X\beta\|_2^2 \\ \text{subject to} \quad & \|\beta\|_1 \leq t \end{aligned}$$

where t is a parameter determining the degree of regularization. Hence, the constrained formulation is a constrained convex optimization problem

$$\min_C \quad \frac{1}{n}\|y - X\beta\|_2^2$$

where

$$C = \left\{\beta \in \mathbb{R}^d : \ \|\beta\|_1 \leq t\right\}.$$

Here, C is a polytope. We may run projected gradient descent to this problem, but it requires projection onto polytope C which amounts to solving a quadratic program. On the other hand, we know that the problem of optimizing a linear function over polytope C is a linear program, for which we have fast solution methods.

4.2 Conditional gradient method: Frank-Wolfe algorithm

Motivated by the issues of projected gradient descent, we consider the conditional gradient method, introduced by Frank and Wolfe in 1956 [FW56]. Named after the author, the conditional gradient method is often referred to as the Frank-Wolfe algorithm. A pseudo-code of the method is given as follows.

Algorithm 2 Frank-Wolfe algorithm

```
Initialize $x_1 \in C$.
for $t = 1, \dots, T-1$ do
    Take $v_t \in \operatorname{argmin}_{v \in C} \nabla f(x_t)^\top v$.
    Update $x_{t+1} = (1-\lambda_t)x_t + \lambda_t v_t$ for some $0 < \lambda_t < 1$.
end for
Return $x_T$.
```

The main component of the conditional gradient method is to compute the direction v_t by solving

$$\min_{v \in C} \nabla f(x_t)^\top v$$

whose objective is a linear function. In particular, when C is a polyhedron, it is just a linear program. This is in contrast to projected gradient descent which has a quadratic objective for each projection step. For this reason, the conditional gradient method is called "projection-free".

Another difference compared to projected gradient descent is that the direction we take for an update can be different from $-\nabla f$. We provide Figure 12.4 for a pictorial description of the update rule. v_t is a point up to which we can move as far as we can in the direction of $-\nabla f(x_t)$ within C.

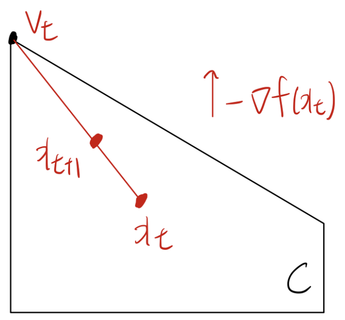

Figure 12.4: Illustration of an update from conditional gradient descent

Then we take a convex combination of the current point x_t and v_t to obtain the new iterate x_{t+1}.

Definition 12.6. We say that a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is β-*smooth* with respect to a norm $\|\cdot\|$ for some $\beta > 0$ if

$$\|\nabla f(x) - \nabla f(y)\|_* \leq \beta \|x - y\|$$

holds for any $x, y \in \mathbb{R}^d$ where $\|\cdot\|_*$ denotes the dual norm of $\|\cdot\|$.

The next theorem shows that conditional gradient descent converges with rate $O(1/T)$ for any smooth function with repsect to an arbitrary norm.

Theorem 12.7. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function that is β-smooth with respect to a norm $\|\cdot\|$ for some $\beta > 0$. Let $\{x_t : t = 1, \ldots, T\}$ be the sequence of iterates generated by the Frank-Wolfe algorithm with*

$$\lambda_t = \frac{2}{t+1}$$

for each t. Then for any $t \geq 2$,

$$f(x_t) - f(x^*) \leq \frac{2\beta R^2}{t+1}$$

where x^* is an optimal solution to $\min_{x\in C} f(x)$ and $R = \sup_{x,y\in C} \|x - y\|$.

Proof. Note that

$$\begin{aligned} f(x_{t+1}) - f(x_t) &\leq \nabla f(x_t)^\top (x_{t+1} - x_t) + \frac{\beta}{2}\|x_{t+1} - x_t\|^2 \\ &= \lambda_t \nabla f(x_t)^\top (v_t - x_t) + \frac{\beta}{2}\|x_{t+1} - x_t\|^2 \\ &\leq \lambda_t \nabla f(x_t)^\top (x^* - x_t) + \frac{\beta}{2}\|x_{t+1} - x_t\|^2 \\ &\leq \lambda_t (f(x^*) - f(x_t)) + \frac{\beta}{2}\|x_{t+1} - x_t\|^2 \end{aligned}$$

where the first inequality is from the β-smoothness of f, the first equality follows from $x_{t+1} = (1 - \lambda_t)x_t + \lambda_t v_t$, the second inequality is due to the definition of $v_t = \operatorname{argmin}_{v\in C} \nabla f(x_t)^\top v$, and the last inequality is by the convexity of f. Since

$$\|x_{t+1} - x_t\| = \lambda_t \|v_t - x_t\| \leq \lambda_t R,$$

it follows that

$$\begin{aligned} f(x_{t+1}) - f(x^*) &\leq (1 - \lambda_t)(f(x_t) - f(x^*)) + \frac{\beta \lambda_t^2 R^2}{2} \\ &= \frac{t-1}{t+1}(f(x_t) - f(x^*)) + \frac{2\beta R^2}{(t+1)^2}. \end{aligned}$$

By this inequality, it follows that

$$f(x_2) - f(x^*) \leq \frac{\beta R^2}{2} \leq \frac{2\beta R^2}{3}.$$

Then by the induction hypothesis,

$$f(x_{t+1}) - f(x^*) \leq \frac{2(t-1) + 2}{(t+1)^2}\beta R^2 = \frac{t}{(t+1)^2} 2\beta R^2 \leq \frac{1}{t+2}\beta R^2,$$

as required. □

4.3 Low-rank matrix completion

Let $A \in \mathbb{R}^{n\times p}$ be a partially observable matrix, of which the missing entries are filled with 0. We assume that even the non-zero entries of A are some noisy observations of the true values.

Such a matrix A arises in movie rating systems, in which case the rows of A correspond to the users and the columns correspond to the list of movies. Hence, n is the number of users and p is the number of movies. Here, one reasonable assumption is that the movie ratings of users depend on a small set of features and criteria. One way to model this is to impose that the true rating matrix A^* satisfies

$$A \cong A^* = UV^\top$$

where U is an $n \times k$ matrix and V is a $p \times k$ matrix with $k < n, p$. By $A^* = UV^\top$, the true rating matrix A^* has rank at most k. Then, to infer the true matrix A^*, we may attempt to solve the following problem.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|A - X\|_F^2 \\ \text{subject to} \quad & \text{rank}(X) \leq k \end{aligned}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, which essentially extends the ℓ_2 norm over vectors to matrices. Here, the low-rank constraint $\text{rank}(X) \leq k$ is non-convex, so we may consider the following problem instead.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|A - X\|_F^2 \\ \text{subject to} \quad & \|X\|_{\text{nuc}} \leq k \end{aligned} \tag{12.1}$$

where $\|\cdot\|_{\text{nuc}}$ denotes the nuclear norm. Here, (12.1) is a constrained convex optimization problem where the constraint set is given by

$$C = \left\{X \in \mathbb{R}^{n\times p} : \ \|X\|_{\text{nuc}} \leq k\right\}.$$

To solve (12.1), the first approach is to apply projected gradient descent over the constraint set C. It is known that projection onto the constraint set C amounts to computing the singular value decomposition of A [DSSSC08].

The second approach is to use the Frank-Wolfe algorithm. GIven a current iterate matrix X_t, the Frank-Wolfe update step considers

$$V_t \in \text{argmin}\left\{\text{Tr}((A - X_t)^\top V) : \ \|V\|_{\text{nuc}} \leq k\right\}.$$

The associated minimization problem is equivalent to computing the top left and right singular vectors of $X_t - A$, for which we may apply the classical power method [JS10]. Here, the power method does not compute the full singular value decomposition, so it runs faster than the projection operation.

References

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3–4):231–357, 2015. 2, 12.1, 12.2, 12.3

[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. 4.3

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. 4.2

[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. 4.3

[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. 3

[Nes03] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003. 2

[Nes04] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, Norwell, 2004. 3

[NY83] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983. 2