1 Outline

In this lecture, we study

- conic duality,
- optimality conditions,
- gradient descent and analysis.

2 Conic duality

In the last lecture, we learned that a *conic program* is an optimization problem defined with a pointed and closed convex cone K with a nonempty interior:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax - b \in K. \end{aligned} \tag{CP}$$

When $K = \mathbb{R}^n_+$, (CP) reduces to linear program

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \geq b. \end{aligned} \tag{LP}$$

We saw that the dual of the linear program (LP) is given by

$$\begin{aligned} \text{maximize} \quad & b^\top y \\ \text{subject to} \quad & A^\top y = c \\ & y \geq 0. \end{aligned} \tag{dual-LP}$$

Let us see how to derive the dual! Note that for any $y \geq 0$ (or $y \in \mathbb{R}^n_+$) and system $Ax \geq b$, we have $y^\top(Ax - b) \geq 0$ because $y \geq 0$ and $Ax - b \geq 0$. Then it follows that

$$y^\top Ax \geq y^\top b.$$

If y further satisfies

$$A^\top y = c,$$

then we have

$$y^\top Ax = c^\top x \geq y^\top b = b^\top y.$$

In summary, if we take $x \in \mathbb{R}^d$ satisfying $Ax \geq b$ and $y \in \mathbb{R}^n$ with $y \geq 0$ and $A^\top y = c$, then $c^\top x$ is always lower bounded by $b^\top y$. Then we can try to find the best possible lower bound by maximizing the value of $b^\top y$, which is precisely what (dual-LP) does!

Following the basic idea behind obtaining the dual linear program, we may obtain and define the dual of the conic program (CP). The *dual cone* of $K \subseteq \mathbb{R}^n$ is defined as

$$K^* = \left\{ y \in \mathbb{R}^n : \ y^\top x \geq 0 \ \ \forall x \in K \right\}.$$

The dual cone of the nonnegative orthant $\mathbb{R}^d_+$ is $\mathbb{R}^d_+$ itself.

Example 4.1. The dual cone of the positive semidefinite cone $\mathbb{S}^d_+$ is given by

$$\left\{ X \in \mathbb{R}^{d\times d} : \ \operatorname{tr}(X^\top S) = \sum_{i=1}^{d}\sum_{j=1}^{d} X_{ij}S_{ij} \geq 0 \ \ \forall S \in \mathbb{S}^d_+ \right\}.$$

In fact, the positive semidefinite cone $\mathbb{S}^d_+$ is *self-dual*, meaning that its dual cone is itself.

Theorem 4.2 (See Theorem 2.3.1 in [BTN01])**.** *Let K be a pointed and closed convex cone with nonempty interior. Then its dual cone K^* is also a pointed and closed convex cone with nonempty interior. Moreover, $(K^*)^* = K$.*

Let us see how to derive and define the dual of the conic program!

(1) Take x such that $Ax - b \in K$ and $y \in K^*$. Then $y^\top(Ax - b) \geq 0$, and therefore,

$$y^\top Ax \geq y^\top b.$$

(2) If $y \in K^*$ further satisfies $A^\top y = c$, then

$$c^\top x = y^\top Ax \geq y^\top b = b^\top y.$$

(3) Then

$$\begin{aligned} \text{maximize} \quad & b^\top y \\ \text{subject to} \quad & A^\top y = c \\ & y \in K^* \end{aligned} \tag{dual-CP}$$

provides a lower bound on the value of (CP). Here, (dual-CP) is the dual conic program of (CP).

Taking the dual of a maximization problem is similar; the dual will give an upper bound on the problem.

Example 4.3. We consider the following semidefinite program.

$$\begin{aligned} \text{maximize} \quad & \sum_{\ell=1}^{m} b_\ell y_\ell \\ \text{subject to} \quad & \sum_{\ell=1}^{m} y_\ell A_\ell \preceq C \end{aligned}$$

To obtain its dual, we take a positive semidefinite matrix X. As the positive semidefinite cone $\mathbb{S}^d_+$ is self-dual, it follows that

$$\operatorname{tr}\left(X^\top \left(C - \sum_{\ell=1}^{m} y_\ell A_\ell \right) \right) = \operatorname{tr}(C^\top X) - \sum_{\ell=1}^{m} y_\ell \cdot \operatorname{tr}((A_\ell)^\top X) \geq 0.$$

If X satisfies

$$\operatorname{tr}((A_\ell)^\top X) = b_\ell \quad \text{for } \ell = 1, \ldots, m,$$

then

$$\operatorname{tr}(C^\top X) \geq \sum_{\ell=1}^{m} y_\ell \cdot \operatorname{tr}((A_\ell)^\top X) = \sum_{\ell=1}^{m} b_\ell y_\ell.$$

This means that

$$\begin{aligned} \text{minimize} \quad & \operatorname{tr}(C^\top X) \\ \text{subject to} \quad & \operatorname{tr}((A_\ell)^\top X) = b_\ell \quad \text{for } \ell = 1, \ldots, m \\ & X \succeq 0 \end{aligned}$$

provides an upper bound on the first semidefinite program.

We have shown that the value of the conic program (CP) is lower bounded by the value of its dual, given by (dual-CP). What is striking is that the two values coincide under some conditions! For the case of linear programming, this is known as the strong LP duality theorem. To be precise, the value of (LP) and that of (dual-LP) are the same if (LP) is feasible and bounded. It turns out that we may extend this strong duality result to general conic programs.

We have already observed that the weak duality for LP extends to conic programming. Formally,

Theorem 4.4. *The optimal value of* (dual-CP) *is a lower bound on that of* (CP).

Before we state the strong duality theorem for conic programming, we need to define the notion of *strict feasibility*.

Definition 4.5. We say that a solution x is *strictly feasible* to (CP) if $Ax - b$ belongs to the interior of K. When that is the case, we write it as

$$Ax - b >_K 0 \quad \text{or} \quad Ax >_K 0.$$

Moreover, we say that (CP) is *strictly feasible* if it has a strictly feasible solution.

For example, x is strictly feasible to (LP) if $Ax > b$. Now we are ready to state the following duality result.

Theorem 4.6 (See Theorem 2.4.1 in [BTN01])**.** *The following statements hold for* (CP) *and* (dual-CP).

1. *If* (CP) *is strictly feasible and bounded, then* (dual-CP) *is solvable and the optimal values of* (CP) *and* (dual-CP) *are the same.*
2. *If* (dual-CP) *is strictly feasible and bounded, then* (CP) *is solvable and the optimal values of* (CP) *and* (dual-CP) *are the same.*

We will learn the notion of *Lagrangian duality* later in the course, and the strict feasibility condition is analogous to the *Slater condition*.

3 Second-order cone programming

A *second-order cone program (SOCP)* is an optimization problem of the following form:

$$\begin{aligned} \text{minimize} \quad & f^\top x \\ \text{subject to} \quad & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i \quad \text{for } i = 1, \ldots, m, \\ & Ex = g. \end{aligned} \tag{SOCP}$$

Exercise 4.7. Prove that (SOCP) is a conic program.

3.1 Example: chance-constrained linear programming

We consider a *chance-constrained program (CCP)* given as follows.

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & \mathbb{P}\left(a^\top x \le b\right) \ge 1-\epsilon \end{aligned} \tag{CCP}$$

where the constraint vector a is random. The problem is to find a solution x satisfying the constraint $a^\top x \le b$ with probability at least $1-\epsilon$. For example, we can formulate the portfolio optimization as (CCP).

$$\begin{aligned} \text{minimize} \quad & p^\top x \\ \text{subject to} \quad & \mathbb{P}\left(r^\top x \ge 1+\alpha\right) \ge 1-\epsilon, \\ & 1^\top x = 1 \end{aligned}$$

where p_i is the unit price of financial asset $i \in [d]$, and r_i is the random return of asset $i \in [d]$. The goal of the problem is to find a portfolio x whose return is at least $1+\alpha$ with probability at least $1-\epsilon$ while minimizing the unit price of the portfolio.

When the coefficient vector a in (CCP) follows the multivariate Gaussian distribution with mean $\bar{a}$ and covariance Σ, we can reformulate it as a second-order cone program. Let $u = a^\top x$. Then u is a random variable with mean $\bar{u} = \bar{a}^\top x$ and variance $\sigma^2 = x^\top \Sigma x$. Then $\mathbb{P}\left(a^\top x \le b\right) \ge 1-\epsilon$ can be rewritten as

$$\mathbb{P}\left(\frac{u-\bar{u}}{\sigma} \le \frac{b-\bar{u}}{\sigma}\right) \ge 1-\epsilon. \tag{4.1}$$

Here,

$$\frac{u-\bar{u}}{\sigma}$$

follows the Gaussian distribution with mean 0 and variance 1 whose cumulative distribution function is given by Φ, i.e.

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2} dt.$$

Then (4.1) is equivalent to

$$\Phi\left(\frac{b-\bar{u}}{\sigma}\right) \ge 1-\epsilon$$

and therefore is equivalent to

$$\frac{b-\bar{u}}{\sigma} = \frac{b-\bar{a}^\top x}{\|\Sigma^{1/2} x\|_2} \ge \Phi^{-1}(1-\epsilon).$$

Hence, (CCP) can be equivalently written as

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & \bar{a}^\top x + \Phi^{-1}(1-\epsilon)\|\Sigma^{1/2} x\|_2 \le b. \end{aligned}$$

4 Optimality Conditions for Convex Minimization

4.1 Local Optimality Implies Global Optimality

A feasible solution x^* is **locally optimal** to the optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array} \tag{P}$$

if there exists $R > 0$ such that

$$f(x^*) = \min \{f(x) : \ x \in C, \ \|x - x^*\| \leq R\}.$$

Theorem 4.8. *Any locally optimal solution to a convex optimization problem is (globally) optimal.*

Proof. Suppose for a contradiction that a locally optimal solution x^* to a convex optimization problem $\min_{x \in C} f(x)$ is not globally optimal. Then there exists $y \in C$ such that $f(y) < f(x^*)$. By the local optimality of x^*, there exists $R > 0$ such that $f(x^*) = \min\{f(x) : x \in C, \ \|x - x^*\| \leq R\}$, which implies that $\|y - x^*\| > R$. Let z be defined as

$$z = x^* + \frac{R}{\|y - x^*\|}(y - x^*) = \left(1 - \frac{R}{\|y - x^*\|}\right) x^* + \frac{R}{\|y - x^*\|} y.$$

Since z is a convex combination of x^* and y, it follows that $z \in C$ and

$$f(z) \leq \frac{R}{\|y - x^*\|} f(y) + \left(1 - \frac{R}{\|y - x^*\|}\right) f(x^*) < f(x^*)$$

However, we have $\|z - x^*\| = R$, contradicting the assumption that $f(x^*) = \min\{f(x) : x \in C, \ \|x - x^*\| \leq R\}$. □

For nonconvex problems, a locally optimal solution is not necessarily an optimal solution, illustrated in Figure 4.1.

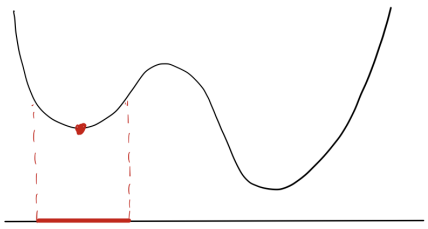

Figure 4.1: Local optimal solution that is not optimal

4.2 First-Order Optimality Condition

Next we establish an optimality condition for convex optimization problems with a differentiable objective.

Theorem 4.9. *For a convex optimization problem of the form (P) with f differentiable, $x^* \in C$ is an optimal solution if and only if*

$$\nabla f(x^*)^\top (x - x^*) \geq 0 \quad \textit{for all } x \in C.$$

Figure 4.2 describes the optimality conditions for functions from $\mathbb{R}^2$ to $\mathbb{R}$. Basically, a solution x^* is optimal if we cannot move further from x^* in C in the direction of decreasing f. If $\nabla f(x^*) = 0$, then x^* is optimal.

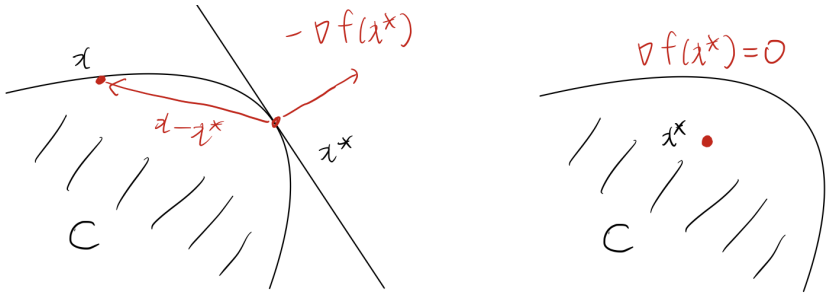

Figure 4.2: Optimality of bi-variate convex functions

By Theorem 4.9, a sufficient condition for optimality is that $\nabla f(x^*) = 0$. This, in fact, is a necessary and sufficient condition for the unconstrained case.

Theorem 4.10. *$x^* \in \mathbb{R}^d$ is optimal to $\min_{x\in\mathbb{R}^d} f(x)$ if and only if*

$$\nabla f(x^*) = 0.$$

Proof. ($\Leftarrow$) If $\nabla f(x^*) = 0$, then it trivially holds that $\nabla f(x^*)^\top (x - x^*) \geq 0$ for $x \in \mathbb{R}^d$. Then x^* is optimal due to Theorem 4.9.

($\Rightarrow$) Let $x = x^* - \alpha\nabla f(x^*)$. Then by Theorem 4.9, we have

$$\nabla f(x^*)^\top (x - x^*) = -\alpha\|\nabla f(x^*)\|_2^2 \geq 0.$$

This in turn implies that $\|\nabla f(x^*)\|_2 = 0$ and thus $\nabla f(x^*) = 0$. □

Figure 4.3 describes the optimality conditions for functions from $\mathbb{R}$ to $\mathbb{R}$.

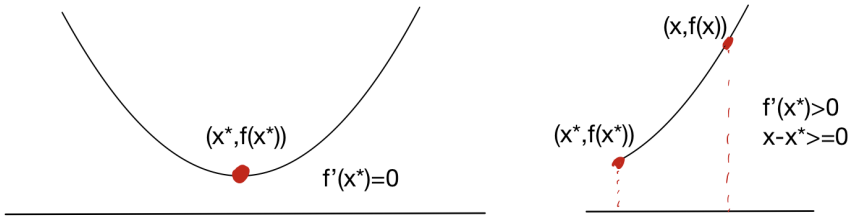

Figure 4.3: Optimality of univariate convex functions

5 Introduction to Gradient Descent

5.1 Generic Descent Method

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function. Given a point $x \in \mathbb{R}^d$, we say that a nonzero vector $d \in \mathbb{R}^d \setminus \{0\}$ is a **descent direction** of f at x if there exists some $\epsilon > 0$ such that

$$f(x + \eta d) < f(x)$$

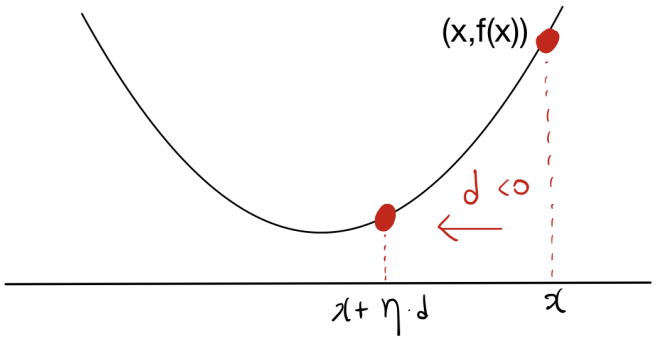

Figure 4.4: Illustration of descent directions

for any $0 < \eta \leq \epsilon$.

Hence, moving towards a descent direction d can decrease the function value, but how much we move along the direction, captured by η, is important. Here, η is referred to as a **step size** or as a **learning rate**. Based on descent directions and proper step sizes, we may develop the following algorithm for minimizing a function.

Algorithm 1 Generic descent method

Initialize $x_1 \in \text{dom}(f)$.
for $t = 1, \ldots, T$ **do**
  Fetch a descent direction d_t.
  $x_{t+1} = x_t + \eta_t d_t$ for a step size $\eta_t > 0$.
end for

Whether the descent method, given by Algorithm 1, converges or not depends on how we choose the step sizes η_t for $t \geq 1$.

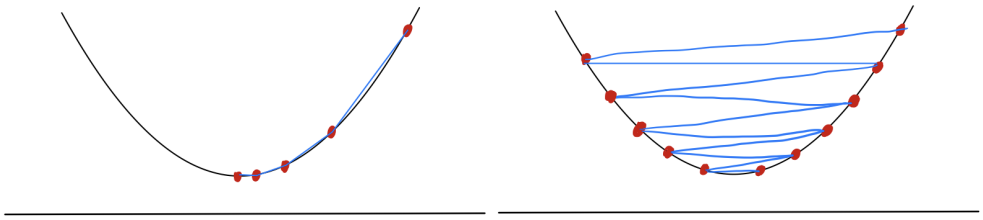

Figure 4.5: Different sequences of step sizes and convergence behavior

We characterize descent directions in terms of the gradient. If f is differentiable, we have

$$\lim_{\eta \to 0+} \frac{f(x+\eta d) - f(x)}{\eta} = d^\top \nabla f(x) \tag{4.2}$$

as the limit exists. Then $\nabla f(x)^\top d$ measures the rate of change in f along direction d at x. Moreover, the following lemma directly follows from (4.2) that holds for differentiable functions.

Lemma 4.11. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. Then a nonzero vector $d \in \mathbb{R}^d \setminus \{0\}$ is a descent direction if*

$$\nabla f(x)^\top d < 0.$$

For example, $-\nabla f(x)$ is a descent direction at any x.

5.2 Gradient Descent

The **steepest direction** of a differentiable function f at a point x can be defined as

$$\arg\min \left\{ \nabla f(x)^\top d : \ \|d\|_2 = 1 \right\} = \left\{ -\frac{1}{\|\nabla f(x)\|_2} \nabla f(x) \right\}.$$

Basically, the steepest direction, which is the direction opposite to the gradient, is the one with the highest rate of decrease of f at x. Then using $-\nabla f$ for a descent direction at any point of the descent method, we obtain the following algorithm, which is commonly known as gradient descent.

Algorithm 2 Gradient descent

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

5.3 Taylor approximation interpretation

Given a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ and a point $x_t \in \mathrm{dom}(f)$, the first-order Taylor approximation of f at x_t is given by

$$f(x) \approx f(x_t) + \nabla f(x_t)^\top (x - x_t).$$

If f is convex, then by the first-order characterization of convexity, we know that the first-order Taylor approximation is a lower bound on f. Moreover, as it provides an approximation of f, we can try to minimize $f(x) \approx f(x_t) + \nabla f(x_t)^\top (x - x_t)$ instead of f. However, the first-order Taylor approximation is a linear function, which means that

$$\min_x \left\{ f(x_t) + \nabla f(x_t)^\top (x - x_t) \right\} = -\infty.$$

Instead of minimizing the first-order Taylor approximation directly, we add a proximity term as follows.

$$f(x) \approx f(x_t) + \nabla f(x_t)^\top (x - x_t) + \underbrace{\frac{1}{2\eta_t} \|x - x_t\|_2^2}_{\text{proximity term}}.$$

When minimizing the second approximation, we cannot choose a solution that is too far from x_t, for otherwise, the corresponding value will be high due to the proximity term. Again, we minimize the approximation instead of f and take a unique minimizer x_{t+1} as follows.

$$x_{t+1} \in \operatorname*{argmin}_x \left\{ f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{1}{2\eta_t} \|x - x_t\|_2^2 \right\}.$$

This gives us an iterative algorithm for minimizing f. Again, the proximity term prevents the solution from being too far away from the initial point x_t. The larger η_t is, the closer the solution x_{t+1} is to the starting point x_t. In fact, there is a closed-form expression for x_{t+1}. Recall that the approximation with the proximity term is differentiable, and therefore, it follows from the optimality condition that

$$\nabla f(x_t) + \frac{1}{\eta_t}(x_{t+1} - x_t) = 0.$$

This is equivalent to

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

which is precisely the gradient descent iteration.

6 Convergence of gradient descent

6.1 Approximate optimality and oracle complexity

It is often difficult to obtain an exact minimizer of the problem. Instead, we seek for an approximately optimal solution. Namely, for a given error tolerance $\epsilon > 0$, the goal is to find a solution x such that

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

We say that a solution x satisfies this condition is ϵ-optimal. One can predict that as ϵ gets smaller, it is harder to find an ϵ-optimal solution. Then, how do we measure how hard it is to find an ϵ-optimal solution? Here, we need some notion of computational complexity.

A computational abstraction for convex optimization is the notion of *oracles*. A *first-order oracle* returns the gradient $\nabla f(x)$ of a given solution x. A *second-order oracle* returns the Hessian $\nabla^2 f(x)$ of a given solution x. A *zeroth-order oracle* returns the objective value $f(x)$ of a given solution x. Here, we say that an algorithm is a *first-order method* if it relies on a first-order oracle. How do we measure the complexity of an algorithm for convex optimization? Basically, we count the number of oracle calls to find an ϵ-optimal solution. For example, the famous gradient descent method finds an ϵ-optimal solution with $O(1/\epsilon^2)$ first-order oracle calls. Here, we often simply say "iterations" instead of "oracle calls".

6.2 Lipschitz continuous functions

We say that a differentiable function is Lipschitz continuous if there exists some $L > 0$ such that

$$|f(x) - f(y)| \leq L\|x - y\|_2$$

for any $x, y \in \mathbb{R}^d$. More precisely, we say that f is L-Lipschitz continuous in the norm $\|\cdot\|_2$. This is equivalent to

$$\|\nabla f(x)\|_2 \leq L$$

for any $x \in \mathbb{R}^d$.

Theorem 4.12. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L-Lipschitz continuous in the ℓ_2-norm and convex, and let $\{x_t : t = 1, \ldots, T\}$ be the sequence of iterates generated by gradient descent with step size*

$$\eta_t = \frac{1}{\sqrt{t}}$$

for each t. Then

$$f\left(\frac{\sum_{t=1}^T \eta_t x_t}{\sum_{t=1}^T \eta_t}\right) - f(x^*) \leq \frac{\|x_1 - x^*\|_2^2}{\sqrt{T}} + \frac{L^2(1 + \log T)}{2\sqrt{T}}$$

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

Proof. Let $\eta_t = 1/\sqrt{t}$.

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

where the inequality follows from $f(x^*) \geq f(x_t) + \nabla f(x_t)^\top (x^* - x_t)$. Then it follows that

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

Summing this over $t = 1, \ldots, T$, we obtain

$$\begin{aligned}
2\sum_{t=1}^T \eta_t \left(f(x_t) - f(x^*)\right) &\leq \|x_1 - x^*\|_2^2 - \|x_{T+1} - x^*\|_2^2 + \sum_{t=1}^T \eta_t^2 \|\nabla f(x_t)\|_2^2 \\
&\leq \|x_1 - x^*\|_2^2 + L^2 \sum_{t=1}^T \eta_t^2.
\end{aligned}$$

Dividing the inequality by $2\sum_{t=1}^T \eta_t$, we deduce that

$$f\left(\frac{\sum_{t=1}^T \eta_t x_t}{\sum_{t=1}^T \eta_t}\right) - f(x^*) \leq \frac{\sum_{t=1}^T \eta_t f(x_t)}{\sum_{t=1}^T \eta_t} - f(x^*) \leq \frac{\|x_1 - x^*\|_2^2}{\sum_{t=1}^T \eta_t} + \frac{L^2 \sum_{t=1}^T \eta_t^2}{2\sum_{t=1}^T \eta_t}$$

where the first inequality is due to convexity of f. As we have

$$\sum_{t=1}^T \frac{1}{\sqrt{t}} \geq \sqrt{T} \quad \text{and} \quad \sum_{t=1}^T \frac{1}{t} \leq 1 + \log T,$$

we obtain

$$f\left(\frac{\sum_{t=1}^T \eta_t x_t}{\sum_{t=1}^T \eta_t}\right) - f(x^*) \leq \frac{\|x_1 - x^*\|_2^2}{\sqrt{T}} + \frac{L^2(1 + \log T)}{2\sqrt{T}},$$

as required. □

References

[BTN01] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization*. Society for Industrial and Applied Mathematics, 2001. 4.2, 4.6