Lecture 2

2.1 Lower Bound for Global Optimization

In this section we establish a *lower complexity bound* on global minimization, that is, the minimal number of iterations required by *any* optimization algorithm from a given class to solve the problem. We will study zeroth-order methods and establish that the grid search algorithm from the previous lecture is *optimal*: thus, its upper complexity bound matches the lower bound, up to a constant.

We consider the following problem:

$$\min_{x \in B} f(x), \tag{2.1}$$

where B is a ball of radius $R > 0$ around origin in an *arbitrary norm* $\|\cdot\|$:

$$B \;\; = \;\; \{x \in \mathbb{R}^n \;:\; \|x\| \leq R\}, \tag{2.2}$$

and $f : B \to \mathbb{R}$ is a Lipschitz continuous function, with constant $L > 0$:

$$|f(y) - f(x)| \;\; \leq \;\; L\|y - x\|, \qquad x, y \in B. \tag{2.3}$$

The goal is to find a point $\bar{x} \in B$ that is an approximate global solution in terms of the functional residual:

$$f(\bar{x}) - f^\star \;\; \leq \;\; \varepsilon, \tag{2.4}$$

for a given $\varepsilon > 0$.

Note that in the previous lecture we fixed the ℓ_∞-norm, which is the simplest for the construction of the grid search. In contrast, our lower bound will work for arbitrary norms, where the choice of norm defines the underlying geometry of the problem.

2.1.1 Packing Problem

Establishing the desired lower bound is closely related to the following famous *packing problem*, which remains an active area of research. We will only need the very basic facts about this problem.

Suppose we have a set of K points in our large ball: $x_1, \ldots, x_K \in B$ and consider a set of small balls, each of radius $0 < r < R$, centered at these points:

$$b_i \;\; = \;\; \{x \in \mathbb{R}^n \;:\; \|x - x_i\| \leq r\}, \qquad 1 \leq i \leq K.$$

We say that the set of balls $\{b_1, \ldots, b_K\}$ is a *packing* in B if

- Each $b_i \subseteq B$;
- The interiors of any two balls are *disjoint*: $\operatorname{int} b_i \cap \operatorname{int} b_j = \varnothing$, for $i \neq j$.

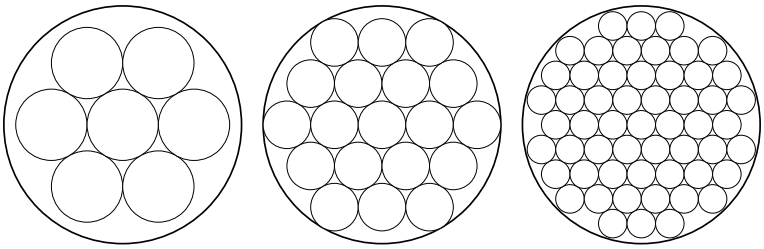

Figure 2.1: Packings of a unit Euclidean ball in $\mathbb{R}^2$.

In other words, we "fill" the large ball B with small balls $\{b_1, \ldots, b_N\}$ without overlaps. We say that packing $\{b_1, \ldots, b_K\}$ is *maximal* if we cannot add a single ball of radius r without it overlapping one of the existing balls. See Figure 2.1 for an illustration. A maximal packing always exists, since any packing can be greedily extended to a maximal one.

We establish the following simple lower bound on the size of a maximal packing.

Proposition 2.1.1. *Let $\{b_1, \ldots b_K\}$ be a maximal packing. Then,*

$$K \;\geq\; \left(\tfrac{R-r}{2r}\right)^n. \tag{2.5}$$

Proof. Let us consider a slightly shrunken ball $\bar{B}$ of radius $R - r > 0$:

$$\bar{B} \;=\; \{x \in \mathbb{R}^n \;:\; \|x\| \leq R - r\},$$

so that for any $x \in \bar{B}$, the small ball of radius r centered at x belongs entirely to B.

Since the packing $\{b_1, \ldots, b_K\}$ is maximal in B, for any point $x \in \bar{B}$, there exists some x_i $(1 \leq i \leq K)$, which is the center of the ball b_i, such that

$$\|x - x_i\| \;<\; 2r.$$

Otherwise, we could place a new ball of radius $r > 0$ centered at x and it would not overlap with any of $\{b_1, \ldots, b_K\}$, which contradicts the maximality of the packing.

Hence, the set of small balls with the same centers but twice the radius,

$$c_i \;=\; \{x \in \mathbb{R}^n \;: \|x - x_i\| \leq 2r\}, \qquad 1 \leq i \leq K,$$

is a *covering* of $\bar{B}$:

$$\bar{B} \;\subseteq\; \bigcup_{1 \leq i \leq K} c_i. \tag{2.6}$$

Taking the volume of the left and the right-hand side in (2.6), we obtain

$$\mathrm{Vol}(\bar{B}) \;\leq\; \sum_{i=1}^{K} \mathrm{Vol}(c_i) \;=\; K \cdot \mathrm{Vol}(c_1). \tag{2.7}$$

Let $\alpha := \mathrm{Vol}(\,\{x \in \mathbb{R}^n \;:\; \|x\| \leq 1\}\,)$ be the volume of the unit ball. Since in $\mathbb{R}^n$, the volume is a homogeneous function of degree n, we get:

$$\mathrm{Vol}(\bar{B}) \;=\; \alpha \cdot (R - r)^n,$$

$$\mathrm{Vol}(c_1) \;=\; \alpha \cdot (2r)^n.$$

Substituting these values in (2.7), cancelling α, and rearranging the terms completes the proof. □

Remark 2.1.2. As in the previous lecture, let us consider the ball in ℓ_∞-norm, which is the box with side length $2R$. If we partition each side into $p \geq 1$ equal parts, where p is an integer parameter, we cover the entire box with $K = p^n$ small boxes, each of radius $r = \frac{R}{p}$ in the ℓ_∞-norm. Clearly, this gives us a maximal packing of size

$$K \;=\; \left(\tfrac{R}{r}\right)^n,$$

which sharpens the general lower bound provided by (2.5).

2.1.2 Resisting Oracle

We would like to establish a lower bound for the complexity of any zeroth-order algorithm applied to a problem from our class (2.1)–(2.4).

To derive the lower bound, it is useful to employ the idea of the so-called *resisting oracle*. We notice that at each iteration of a method, the oracle needs to return only the value of the objective function at a given point. It does not control the requested point, but it *controls the answer*. Therefore, the resisting oracle may always return the information that is least useful to the algorithm, forcing the algorithm to run as long as possible.

When playing this game, the resisting oracle must ensure only that in the end, when the algorithm returns a result, the oracle is able to *reveal at least one function* belonging to the problem class that is consistent with all previous answers.

In our current case, the resisting oracle is very simple:

$$\boxed{\textbf{Resisting oracle:}\ \text{always return } \mathcal{O}(x) \equiv \{0\}.} \tag{2.8}$$

Therefore, for every point that a method requests the function value, the function value will be zero. We can assume, without loss of generality, that the result of an algorithm after $k \geq 0$ is always the *last requested point* x_k, and thus $f(x_k) = 0$.

At the same time, after the method returns the result, we are able to construct a function which is consistent with all requested points, and that has $f^\star < 0$.

We denote by $K(R, r, n)$ a lower bound for the size of any maximal packing of n-dimensional ball B of radius R in a given norm $\|\cdot\|$, by small balls of radius $r < R$. From Proposition 2.1.1 we can take at least

$$K(R, r, n) \;=\; \left\lceil \left(\tfrac{R-r}{2r}\right)^n \right\rceil. \tag{2.9}$$

We use the lower bound on a maximal packing to prove the following result.

Proposition 2.1.3. *Let $0 < r < R$ be fixed. Consider any zeroth-order algorithm running for $k < K(R, r, n) - 1$ iterations. Then, there exists a Lipschitz continuous function f with Lipschitz constant $L > 0$, such that the result x_k of the algorithm when applied to this function satisfies:*

$$f(x_k) - f^\star \;=\; Lr. \tag{2.10}$$

Proof. Let $\{x_0, \ldots, x_k\} \subset B$ be the points generated by the algorithm when interacting with the resisting oracle (2.8). Thus, the result of the algorithm is $f(x_k) = 0$.

Now, let us pick an arbitrary maximal packing of B by balls of radius r. Its size is at least $K(R, r, n)$. Since $k+1 < K(R, r, n)$, there must exist at least one ball in the packing whose interior does not contain any of the points $\{x_0, \ldots, x_k\}$. We denote the center of such a ball by $x^\star$.

Introduce the function

$$f(x) \;=\; L \cdot \min\{0, \|x - x^\star\| - r\},$$

which possesses the following properties:

1. $f^\star = f(x^\star) = -Lr$.

2. For any x such that $\|x - x^\star\| \geq r$, $f(x) \equiv 0$. Hence, this function is consistent with outputs of the resisting oracle.

3. f is Lipschitz continuous with constant $L > 0$.

To check the last property, we need to verify (2.3), for any $x, y \in B$. Consider first the situation when both: $\|x - x^\star\| \geq r$ and $\|y - x^\star\| \geq r$. Then,

$$f(y) - f(x) \;=\; 0 \;\leq\; L\|y - x\|.$$

Now, assume that $\|x - x^\star\| \leq r$ and the other point y is arbitrary. Then,

$$\begin{aligned} f(y) - f(x) &= L \cdot \min\{0, \|y - x^\star\| - r\} - L \cdot (\|x - x^\star\| - r) \\ &\leq L \cdot (\|y - x^\star\| - \|x - x^\star\|) \;\leq\; L\|y - x\|, \end{aligned}$$

where we used triangle inequality in the last bound.

Therefore, f satisifes all the required properties, and (2.10) holds. □

2.1.3 Lower Bound

To establish the lower complexity bound, it remains to calibrate the radius $r > 0$ of the small ball according to the desired accuracy of solving the problem. Combining all elements, we arrive at the following result.

Theorem 2.1.4. *Let parameters $L > 0$, $R > 0$ and $\varepsilon > 0$ be fixed and assume that the target accuracy is sufficiently small: $\varepsilon < \frac{LR}{2}$. Then, the complexity K of any zeroth-order method on our problem class is bounded by*

$$K \;\geq\; \left\lceil \left(\tfrac{LR}{4\varepsilon} - \tfrac{1}{2}\right)^n \right\rceil - 1. \tag{2.11}$$

Proof. We set $r := \frac{2\varepsilon}{L} < R$ and assume that the method runs for $k < K(R, r, n) - 1$ iterations on any function from our problem class, where

$$K(R, r, n) \overset{(2.9)}{=} \left\lceil \left(\tfrac{R-r}{2r}\right)^n \right\rceil \;=\; \left\lceil \left(\tfrac{LR}{4\varepsilon} - \tfrac{1}{2}\right)^n \right\rceil.$$

Then, using Proposition 2.1.3, we conclude that there exist at least one function on which

$$f(x_k) - f^\star \;=\; Lr \;=\; 2\varepsilon,$$

which contradicts that the algorithm finds ε-solution. Hence $k \geq K(R, r, n) - 1$. □

We see that, up to numerical constants, the lower complexity bound of zeroth-order methods on our problem class is

$$K \;=\; \Omega\Big(\Big[\tfrac{LR}{\varepsilon}\Big]^n\Big). \tag{2.12}$$

At the same time, we saw in the previous lecture (Theorem 1.3.1) that for the ℓ_∞-norm, this matches the complexity upper bound $O([\frac{LR}{\varepsilon}]^n)$ achieved by the grid search algorithm. This implies that the grid search is the *optimal method*, and in general, we cannot come up with a faster algorithm.

These news is rather pessimistic, as bound (2.12) is extremely large due to exponential dependence on dimension. Indeed, choosing $L = R = 1$, a very moderate accuracy $\varepsilon = 10^{-2}$, and $n \geq 50$, we obtain from (2.12) that

$$K \gtrsim 10^{100},$$

which is believed to be larger than the number of atoms in the observable universe. Therefore, from the computational perspective, it is impossible to solve the global optimization problem (high-dimensional and non-convex). Instead, we will first focus on a less ambitious goals: finding stationary points for smooth problems.

2.2 Using Local Information: Differentiable Functions

In this section, we review well-known facts about differentiable functions, and discuss how to compute gradients and Hessians. Gradients are very important for optimization: they are used first as a primary search direction for optimization algorithms, and second as optimality conditions for solutions.

2.2.1 On Vector Spaces

We work with a finite-dimensional real vector space, typically denoted by $\mathbb{R}^n$. We denote by $\langle \cdot, \cdot \rangle$ the *standard inned product* for vectors in $\mathbb{R}^n$, for $x, y \in \mathbb{R}^n$: $\langle x, y \rangle = x^\top y = \sum_{i=1}^n x^{(i)} y^{(i)}$.

Sometimes decision variables can be more structured, such as *matrices* or *symmetric matrices*, or combinations of those, for example, all parameters of a neural network grouped by layers. Of course, every matrix or tensor can be reshaped into a vector, thus it is enough to be able to work with vectors. However, even though such a reshape is possible, in practice it is often convenient to keep the initial shape of objects, as for example, for symmetric matrices.

Exercise 2.2.1. Consider the space of symmetric matrices $\mathbb{S}^d = \{X \in \mathbb{R}^{d \times d} : X = X^\top\}$. Construct an explicit basis for $\mathbb{S}^d$ and prove that its dimension is $n = \frac{d(d+1)}{2}$.

For two matrices of the same size $X, Y \in \mathbb{R}^{n \times m}$, the inner product is $\operatorname{tr}(X^\top Y)$. It is immediate to check that this coincides with the standard inner product for vectors, viewed as if we were to reshape matrices into vectors in $\mathbb{R}^{nm}$:

$$\langle X, Y \rangle = \operatorname{tr}(X^\top Y) = \sum_{i=1}^n \sum_{i=1}^m X^{(i,j)} Y^{(i,j)}. \tag{2.13}$$

For the space of symmetric matrices $\mathbb{S}^d$, we use the same inner product induced by $\mathbb{R}^{d \times d}$, namely $\langle X, Y \rangle = \operatorname{tr}(XY)$. Note that it is *not equivalent* to forming two vectors in $\mathbb{R}^{d(d+1)/2}$ and taking their standard inner product, although that would also be a valid choice of inner product on $\mathbb{S}^d$. However, it is often more convenient to use the induced inner product (2.13) for symmetric matrices, as we will do in this course.

For a symmetric matrix $A \in \mathbb{S}^n$, we say that it is *positive-semidefinite* (notation: $A \succeq 0$) if

$$\langle Ah, h \rangle \geq 0, \qquad \forall h \in \mathbb{R}^n.$$

We say that A is *positive-definite* (notation: $A \succ 0$) if

$$\langle Ah, h \rangle > 0, \qquad \forall h \in \mathbb{R}^n \setminus \{0\}.$$

For two symmetric matrices $A, B \in \mathbb{S}^n$, we say $A \succeq B \Leftrightarrow A - B \succeq 0$.

It is known that for a symmetric matrix, all eigenvalues are *real* numbers, which we denote by $\lambda_1(A) \geq \ldots \geq \lambda_n(A)$. It holds: $A \succeq 0 \;\Leftrightarrow\; \lambda_i(A) \geq 0$ for all $1 \leq i \leq n$.

2.2.2 Derivatives

Consider $f : \mathbb{R}^n \to \mathbb{R}^m$. By definition, f is called *differentiable* at $x \in \mathbb{R}^n$ if there exists a linear operator $L : \mathbb{R}^n \to \mathbb{R}^m$ such that

$$f(x+h) \;=\; f(x) + L[h] + o(\|h\|) \quad \Leftrightarrow \quad \lim_{\|h\|\to 0} \tfrac{\|f(x+h)-f(x)-L[h]\|}{\|h\|} \;=\; 0. \tag{2.14}$$

Since in finite-dimensional spaces all norms are topologically equivalent, it does not matter which norm to pick in the definition. It is easy to check that if such a linear operator L exists, then it is unique. It is called the *derivative* of f at x.

Commonly used **notations** for this linear operator are:

$$Df(x) \;\equiv\; df(x) \;\equiv\; f'(x) \;\equiv\; L.$$

In these notes, we will use $Df(x)$ to denote the derivative.

Thus, the derivative is the *best local approximation* of a function f at x by a linear function:

$$f(x+h) \;\approx\; f(x) + Df(x)[h].$$

Note that Df has two "arguments": x and h, and it is linear in h, but not in x.

Example 2.2.1 (Univariate Functions)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be a univariate function. Then, $Df(x)[h] \equiv f'(x) \cdot h \in \mathbb{R}$, for $f'(x) \in \mathbb{R}$, $h \in \mathbb{R}$. In univariate calculus, h is sometimes denoted by 'dx'. Then, $f'(x)dx$ is called the *differential* of the function, $df = f'(x)dx$.

Example 2.2.2 (Squared Euclidean Norm)**.** Let $f(x) = \frac{1}{2}\|x - x_0\|_2^2 = \frac{1}{2}\langle x - x_0, x - x_0\rangle$. Then,

$$\begin{aligned} f(x+h) \;&=\; \tfrac{1}{2}\|x - x_0 + h\|^2 \;=\; \tfrac{1}{2}\|x - x_0\|^2 + \langle x - x_0, h\rangle + \tfrac{1}{2}\|h\|^2 \\ &=\; f(x) + \langle x - x_0, h\rangle + o(\|h\|). \end{aligned}$$

Therefore, $Df(x)[h] = \langle x - x_0, h\rangle$.

Example 2.2.3 (Frobenious Norm)**.** Let $f(X) = \frac{1}{2}\|X\|_F^2 = \frac{1}{2}\operatorname{tr}(X^\top X)$. Then,

$$\begin{aligned} f(X+H) \;&=\; \tfrac{1}{2}\operatorname{tr}((X+H)^\top(X+H)) \;=\; \tfrac{1}{2}\operatorname{tr}(X^\top X) + \operatorname{tr}(X^\top H) + \tfrac{1}{2}\operatorname{tr}(H^\top H) \\ &=\; f(X) + tr(X^\top H) + o(\|H\|). \end{aligned}$$

Therefore, $Df(X)[H] = \operatorname{tr}(X^\top H)$.

Gradients. In optimization, we work with functions that take real values: $f : \mathbb{R}^n \to \mathbb{R}$. Then, the *gradient* of f at x is a *unique vector* $\nabla f(x) \in \mathbb{R}^n$ such that

$$Df(x)[h] \;\equiv\; \langle \nabla f(x), h\rangle. \tag{2.15}$$

Note that the derivative $Df(x)$ on the left-hand side of (2.15) does not depend on a coordinate system, as our definition of the derivative is *coordinate-free*. Conversely, the right-hand side (2.15) depends on the choice of inner product. Hence, the gradient $\nabla f(x)$ *depends on a choice of the coordinate system*, and will change if we change the basis.

Example 2.2.4. From the previous examples, we immediately see that for $f(x) = \frac{1}{2}\|x - x_0\|_2^2$, the gradient is $\nabla f(x) = x - x_0$. For the matrix function $f(X) = \frac{1}{2}\|X\|_F^2$, the gradient is $\nabla f(X) = X$.

Directional Derivative. Let $f : \mathbb{R}^n \to \mathbb{R}$. For any $h \in \mathbb{R}^n$, the *directional derivative* of f at point x *along direction* h is the derivative of the univariate function $\varphi(t) = f(x + th)$ at zero:

$$\frac{\partial f(x)}{\partial h} \quad := \quad \varphi'(0) \quad = \quad \lim_{t \to 0} \frac{f(x+th)-f(x)}{t}.$$

Directional derivative is a weaker notion that differentiability of a function. Even if a function has directional derivatives along any direction, it can be non-differentiable. However, if the function is differentiable and we know the derivative Df, it is very easy to compute the directional derivative.

Proposition 2.2.5. *For a differentiable function, it holds:*

$$\frac{\partial f(x)}{\partial h} \quad = \quad Df(x)[h] \quad = \quad \langle \nabla f(x), h \rangle. \tag{2.16}$$

Exercise 2.2.2. Check (2.16).

Therefore, we have *two principal ways* of computing the gradients:

1. *Coordinate-wise way.* We first compute all partial derivatives $\frac{\partial f(x)}{\partial x^{(i)}}$ for all coordinate directions $e_1, \ldots, e_n \in \mathbb{R}^n$. Then, we combine them into vector:

$$\nabla f(x) \quad = \quad \left(\frac{\partial f(x)}{\partial x^{(1)}}, \ldots, \frac{\partial f(x)}{\partial x^{(n)}} \right)^\top \quad \in \quad \mathbb{R}^n. \tag{2.17}$$

   Theoretically, this approach is completely fine. Moreover, formula (2.17) can be used as the definition of the gradient. However, it can be computationally hard in practice, especially when working with matrix spaces.

2. *Coordinate-free way.* Think of $\nabla f(x)$ as of the *derivative representation*. We first compute the linear operator $Df(x)[h]$, as applied to an arbitrary direction h, and then find the unique vector $\nabla f(x)$ from the following equation: $Df(x)[h] \equiv \langle \nabla f(x), h \rangle$. Note that from linear algebra we know that such representation is always possible. Often, this approach of computing the gradient is much easier.

Note that $\nabla f(x)$ has always *the same shape* as the target variable x (e.g. a vector, a matrix, multiple tensors — layers in neural networks, etc.)

Second Derivative. Assume that f is differentiable at *every point* x, and denote the new function $g(x) := Df(x)[h]$, for some fixed direction h. By definition, the derivative of g at x is defined by

$$g(x + u) \quad = \quad g(x) + Dg(x)[u] + o(\|u\|),$$

and Dg is called the *second derivative* of f. It is denoted by

$$D^2 f(x)[h, u] \quad \equiv \quad Dg(x)[u].$$

When f is sufficiently smooth (e.g. when $D^2 f(x)$ is continuous), it can be shown that the second derivative is symmetric: $D^2 f(x)[h, u] \equiv D^2 f(x)[u, h]$ and linear with respect to both "h" and "u".

A fundamental result from calculus is the following:

Proposition 2.2.6 (Taylor's Formula)**.** *For a twice differentiable function, it holds*

$$f(x + h) \quad = \quad f(x) + Df(x)[h] + \tfrac{1}{2} D^2 f(x)[h, h] + o(\|h\|^2).$$

Hessian Matrix. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function. Then, the *Hessian* of f at x is a unique matrix $\nabla^2 f(x) \in \mathbb{R}^{n\times n}$ such that

$$D^2 f(x)[h,u] \;\equiv\; \langle \nabla^2 f(x)h, u\rangle.$$

Therefore, the Hessian $\nabla^2 f(x)$ is the *representation* of the bilinear symmetric form $D^2 f(x)$, and it depends on the choice of the coordinate system. Since $D^2 f(x)$ is symmetric form, the Hessian is a symmetric matrix: $\nabla^2 f(x) \in \mathbb{S}^n$. Its entries can be computed coordinate-wise as:

$$[\nabla^2 f(x)]^{(i,j)} \;=\; \frac{\partial^2 f(x)}{\partial x^{(i)} \partial x^{(j)}}, \qquad 1 \le i,j \le n.$$

2.2.3 Optimality Conditions

Theorem 2.2.7 (Optimality Conditions)**.** *Let $x^\star$ be a local minimum of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. Then*

$$\nabla f(x^\star) \;=\; 0 \tag{2.18}$$

If function f is twice continuously differentiable, then additionally we have:

$$\nabla^2 f(x^\star) \;\succeq\; 0. \tag{2.19}$$

Proof. Assume $\nabla f(x^\star) \neq 0$. Take $h := -\alpha \nabla f(x^\star)$ with $\alpha > 0$, and consider

$$\begin{aligned} f(x^\star + h) &= f(x^\star) + \langle \nabla f(x^\star), h\rangle + o(\|h\|) \\ &= f(x^\star) - \alpha \|\nabla f(x^\star)\|_2^2 + o(\alpha). \end{aligned}$$

For sufficiently small α, we get: $f(x^\star + h) \le f(x^\star) - \frac{\alpha}{2}\|\nabla f(x^\star)\|_2^2 < f(x^\star)$, which contradicts that $x^\star$ is a local minimum. Thus, we have proved (2.18).

To prove (2.19), we use Taylor's formula:

$$\begin{aligned} f(x^\star + h) &= f(x^\star) + \langle \nabla f(x^\star), h\rangle + \tfrac{1}{2}\langle \nabla^2 f(x^\star)h, h\rangle + o(\|h\|^2) \\ &\overset{(2.18)}{=} f(x^\star) + \tfrac{1}{2}\langle \nabla^2 f(x^\star)h, h\rangle + o(\|h\|^2). \end{aligned}$$

Assume $\nabla^2 f(x^\star) \not\succeq 0$. Then, there exists a direction u such that $\xi = \langle \nabla f(x^\star)u, u\rangle < 0$. Choose $h := \alpha u$ with $\alpha > 0$. We get, for sufficiently small α:

$$f(x^\star + h) \;=\; f(x^\star) + \tfrac{\alpha^2}{2}\xi + o(\alpha^2) \;\le\; f(x^\star) + \tfrac{\alpha^2}{4}\xi \;<\; f(x^\star),$$

which contradicts that $x^\star$ is a local minimum. Thus, (2.19) is true. □

Positive definitness of the Hessian serves as a *sufficient condition* for a point to be a strict local minimum.

Exercise 2.2.3. Let $\bar{x} \in \mathbb{R}^n$ and assume that it holds: $\nabla f(\bar{x}) = 0$ and $\nabla^2 f(\bar{x}) \succ 0$. Then, $\bar{x}$ is a strict local minimum of f, i.e. there exists a neighborhood $U \subset \mathbb{R}^n$ of $\bar{x}$ such that

$$f(\bar{x}) \;<\; f(x), \qquad x \in U \setminus \{\bar{x}\}.$$