Contents

Chapter 1

Nonlinear Optimization

In this chapter, we mainly focus on:

- Main notations and concepts used in *Continuous Optimization*.
- Complexity Analysis of the problems of Global Optimization.
- Local Minimization and 2 main methods: *the Gradient Method* and *the Newton Method*.
- Some standard methods in General Nonlinear Optimization.

1.1 The World of Nonlinear Optimization

1.1.1 General Formulation of the Problem

Let $\boldsymbol{x}$ be an n-dimensional *real vector*

$$\boldsymbol{x} = \left(x^{(1)}, \ldots, x^{(n)}\right)^\top \in \mathbb{R}^n$$

and $f_0(\cdot), \ldots, f_m(\cdot)$ be some *real-valued* functions defined on a set $Q \subset \mathbb{R}^n$. In this way, let's consider the general minimization problem:

$$\begin{aligned} &\min\ f_0(\boldsymbol{x}), \\ &\text{s.t. } f_j(\boldsymbol{x}) \ \& \ 0, \quad j = 1, \ldots, m, \\ &\boldsymbol{x} \in Q, \end{aligned} \tag{1.1}$$

where the sign & can be $\leq$, $\geq$ or $=$.

Notations:

- f_0 is called *objective* function.
- The vector function $\boldsymbol{f}(\boldsymbol{x}) = \left(f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})^\top\right)$ is called the vector of *functional constraints*.
- The set Q is called the *basic feasible set*.
- The set $\mathscr{F} = \{x \in Q \mid f_j(\boldsymbol{x}),\ j = 1, \ldots, m\}$ is called *the entire feasible set* of problem 1.1.

Classification:

1. **Natural Classification**:
   - *Constrained problems*: $\mathscr{F} \subset \mathbb{R}^n$.

- *Unconstrained problems*: $\mathscr{F} \equiv \mathbb{R}^n$.
- *Smooth problems*: all $f_j(\cdot)$ are differentiable.
- *Nonsmooth problems*: there are several nondifferentiable components $f_k(\cdot)$.
- *Linearly constrained problems*: the functional constraints are affine:

$$f_j(x) = \langle \boldsymbol{a}_j, \boldsymbol{x} \rangle + b_j$$

  - *Linear optimization Problem*: $f_0(\cdot)$ is also affine.
  - *Quadratic optimization problem*: $f_0(\cdot)$ is Quadratic.
  - *Quadratic constrained quadratic problem*: $f_0(\cdot), \ldots, f_m(\cdot)$ are all quadratic.

2. **Based on the Feasible Set**:

- Problem 1.1 is called *feasible* if $\mathscr{F} \neq \emptyset$.
- Problem 1.1 is called *strictly* feasible if there exists an $\boldsymbol{x} \in Q$ such that $f_j(\boldsymbol{x}) < 0$ for all inequality constraints and $f_j(\boldsymbol{x}) = 0$ for all equality constraints. (*Slater condition.*)

3. **Based on Solution**:

- A point $\boldsymbol{x}^* \in \mathscr{F}$ is called the optimal *global solution* to problem 1.1 if $f_0(\boldsymbol{x}^*) \leq f_0(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathscr{F}$ (*global minimum*). $f_0(\boldsymbol{x}^*)$ is called the global *optimal value* of the problem.
- A point $\boldsymbol{x}^* \in \mathscr{F}$ is called a *local solution* to problem 1.1 if there exists a set $\hat{\mathscr{F}} \subset \mathscr{F}$ such that $\forall \boldsymbol{x} \in \text{int}\hat{\mathscr{F}},\ f_0(\boldsymbol{x}^*) \leq f_0(\boldsymbol{x})$. If $\forall \boldsymbol{x} \in \hat{\mathscr{F}} \setminus \{\boldsymbol{x}^*\},\ f_0(\boldsymbol{x}^*) \leq f_0(\boldsymbol{x})$, then $\boldsymbol{x}^*$ is called *strict* (or *isolated*) local minimum.

Nonlinear Optimization is very import and promising application theory. It covers almost ALL needs of Operation Research and Numerical Analysis. However, in general, optimization problems should be UNSOLVABLE. It is difficult to believe in the existence of a universal tool which is able to solve all problems in the world.

Summerization

- First we learn the general formulation of optimization problem in 1.1 and related concepts.
- We can also classify the optimization problems into different categories. The classification methods include:
  - Natural Classification.
  - Based on Feasible Set.
  - Based on Solution.

1.1.2 Performance of Numerical Methods

Usually we focus on the best method for a *class* of problem $\mathscr{P} \ni P$. The *performance* of a method, $\mathscr{M}$ on the whole class $\mathscr{P}$ can be a natural measure of its efficiency. Here we assume that the method $\mathscr{M}$ does not have *complete* information about a particular problem P.

- **Model**: The *model* is the *known* (to a numerical scheme) "part" of problem P and is denoted by Σ. The model consists of:

  - The formulation of the problem.
  - The description of the classes of functional components.
  - etc.

- **Oracle**: The oracle is used to describe the process of collecting this data. A oracle $\mathcal{O}$ is just a unit which answers the successive questions of the methods.

The method $\mathscr{M}$ is trying to solve the problem P by collecting and handling the answers.

For each problem we can develop different types of oracles. But let us fix Σ and $\mathcal{O}$. In this case, it is natural to define the performance of $\mathscr{M}$ on $(\Sigma, \mathcal{O})$ as its performance on the worst P_w from $(\Sigma, \mathcal{O})$. Note that P_w can be only bad for $\mathscr{M}$.

Definition 1.1.1 (*Performance*)

The performance of $\mathscr{M}$ on P is the total amount of computational effort *required by the method $\mathscr{M}$ to solve the problem P.*

Notes:

- Solving the problem means finding an *approximate solution* to $\mathscr{P}$ with some accuracy $\epsilon > 0$.
- We use $\mathscr{T}_\epsilon$ to represent a stopping criterion.

Now we have a formal description of the problem class:

$$\mathscr{P} \equiv (\Sigma, \mathcal{O}, \mathscr{T}_\epsilon)$$

In order to solve a problem P from $\mathscr{P}$, we apply it to an *iterative process*.

Algorithm 1: General Iterative Scheme

Input : Starting point x_0 and accuracy $\epsilon > 0$.
Output: Solution $\bar{x}$.
Initialization: Set $k = 0$, $\mathscr{I}_{-1} = \emptyset$. Here k is the iteration counter and $\mathscr{I}_k$ is the accumulated *informational set*.
while *True* **do**
  Call oracle $\mathcal{O}$ at point $\boldsymbol{x}_k$.
  Update the informational set: $\mathscr{I}_k = \mathscr{I}_{k-1} \cup (\boldsymbol{x}_k, \mathcal{O}(x_k))$.
  Apply the rules of method $\mathscr{M}$ to $\mathscr{I}_k$ and generate a new point $\boldsymbol{x}_{k+1}$.
  if $\mathscr{I}_\epsilon$ **then**
    Form output $\bar{\boldsymbol{x}}$.
  else
    $k := k + 1$.
  end
end

Now we can specify the meaning of *computational effort* in our definition of performance. In Algorithm 1, we can see two potentially expensive steps:

- In Step 1, where we call the oracle.
- In Step 3, where we form a new test point.

So, we can introduce two measures of complexity of problem P for method $\mathscr{M}$:

Definition 1.1.2 (*Computational Complexity*)

Analytical complexity: *The number of calls of the oracle which is necessary to solve the problem P up to the accuracy ϵ.*
Arithmetical complexity: *The total number of arithmetic operations (including the work of oracle and work of method), which is necessary for solving problem P up to accuracy ϵ*

Actually, the second one is more realistic. However, for a particular method $\mathscr{M}$ as applied to problem P, arithmetical complexity can be easily obtained from the analytical complexity and complexity of the oracle.

There is one standard assumption on the oracle which allows us to obtain the majority of results on analytical complexity for optimization schemes. This assumption, called *Local Black Box Concept*, is as follows.

Assumption 1.1.1 (*Local Black Box*)

1. *The only information available for the numerical scheme is the* answer of the oracle.
2. *The oracle is* local*: A small variation of the problem far enough from the test point x, which is compatible with the description of the problem class, does not change the answer at x.*

In Assumption 1.1.1, the first one seems like the artificial wall between the method and the oracle. Although it seems natural to give methods full access to the internal structure of the problem, we can find that when problems have a complicated or implicit structure, this access is almost useless.

Here we conclude problem 1.1 as a *functional model* of optimization problem. According to the degree of smoothness, we can apply different types of oracle:

- Zero-order Oracle: returns the function value $f(\boldsymbol{x})$.
- First-order Oracle: returns the function value $f(\boldsymbol{x})$ and the gradient $\boldsymbol{\nabla} f(\boldsymbol{x})$.
- Second-order Oracle: returns $f(\boldsymbol{x})$, $\boldsymbol{\nabla} f(\boldsymbol{x})$, and the Hessian $\boldsymbol{\nabla}^2 f(\boldsymbol{x})$

Summerization

- Usually we focus on the performance of method $\mathscr{M}$ on a problem class $\mathscr{P}$, rather than a particular problem. There are also 2 import concepts about problem: *model* and *oracle*.
- The performance of method $\mathscr{M}$ on a problem class $\mathscr{P}$ is defined as the total amount computational effort required by the method $\mathscr{M}$ to solve the problem P, just as Definition 1.1.1.
  - Computational complexity: *Analytical* and *Arithmetical*, just as Definition 1.1.2.
  - Solving the problem: finding an *approximate solution* with accuracy $\epsilon > 0$.
- We can use *General Iterative Scheme* in Algorithm 1 to deal with most of optimization problems.
- Standard assumption on oracle: *Local Black Box Assumption* in Assumption 1.1.1.

1.1.3 Complexity Bounds for Global Optimization

Let's consider *n-dimensional box problem*:

$$\begin{aligned} &\min_{x \in B_n} \; f(\boldsymbol{x}) \\ &B_n = \{\boldsymbol{x} \in \mathbb{R}^n \mid 0 \le x^{(i)} \le 1, \; i = 1, \ldots, n\} \end{aligned} \tag{1.2}$$

In our terminology, this is a constrained minimization problem without functional constraints. B_n is the basic feasible set, which can be seemed as a n-dimensional box.

Here we use l_∞-norm to measure distances:

$$\|\boldsymbol{x}\|_\infty = \max_{1 \le i \le n} \left| x^{(i)} \right|$$

Then we make the assumption:

Assumption 1.1.2

the objective function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ *is* Lipschitz continuous *on* B_n*:*

$$|f(\boldsymbol{x} - \boldsymbol{y})| \le L \|\boldsymbol{x} - \boldsymbol{y}\|_\infty \quad \forall \boldsymbol{x}, \boldsymbol{y} \in B_n,$$

with some constant L *(*Lipschitz constant*).*

Let us consider a very simple method for solving problem 1.2, which is called *Uniform Grid Method* $\mathscr{G}(p)$.

Algorithm 2: Uniform Grid Method $\mathscr{G}(p)$

Input : $p \ge 1$

Output: The minimal pair $(\bar{\boldsymbol{x}}, f(\bar{\boldsymbol{x}}))$.

Form p^n points

$$\boldsymbol{x}_\alpha = \left(\frac{2i_1 - 1}{2p}, \frac{2i_2 - 1}{2p}, \ldots, \frac{2i_n - 1}{2p} \right)^\top.$$

where $\alpha \equiv (i_1, \ldots, i_n) \in \{1, \ldots, p\}^n$.

Among all points $\boldsymbol{x}_\alpha$, find the point $\bar{\boldsymbol{x}}$ with the minimal value of the objective function.

Thus, this method forms a uniform grid of the test points in the n-dimensional box, computes the best value of the objective function over the grid, and returns this value as an approximate solution to problem 1.2. In our terminology, this is a zero-order iterative method without any influence from the accumulated information on the sequence of test points. Then let's focus on its efficiency estimate.

Theorem 1.1.1

Let f^* *be a global optimization value of problem 1.2. Then*

$$f(\bar{\boldsymbol{x}}) - f^* \le \frac{L}{2p}$$

Proof of Theorem 1.1.1 : For a multi-index $\alpha = (i_1, \ldots, i_n)$, define

$$X_\alpha = \left\{ \boldsymbol{x} \in \mathbb{R}^n \middle| \|\boldsymbol{x} - \boldsymbol{x}_\alpha\|_\infty \le \frac{1}{2p} \right\}$$

Obviously, $\bigcup_{\alpha \in \{1, \ldots, p\}^n} X_\alpha = B_n$.

Let $\boldsymbol{x}^*$ be a global solution of our problem. Then there exists a multi-index α^* such that $\boldsymbol{x}^* \in X_{\alpha^*}$. Note that $\|\boldsymbol{x}^* - \boldsymbol{x}_{\alpha^*}\|_\infty \leq 1/2p$. Therefore,

$$f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}_{\alpha^*}) - f(\boldsymbol{x}^*) \leq \frac{L}{2p}$$

■

Let us conclude with the definition of our problem class. We fix our goal as follows:

$$\text{Find } \bar{\boldsymbol{x}} \in B_n : \quad f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*) \leq \epsilon \tag{1.3}$$

Corollary 1.1.1

According to Assumption 1.1.2 and Theorem 1.1.1, the analytical complexity of problem class 1.2 for method $\mathscr{G}$ is at most

$$\mathscr{A}(\mathscr{G}) = \left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor + 1 \right)^n .$$

Proof of Corollary 1.1.1 : Take $p = \left\lfloor \frac{L}{2\epsilon} \right\rfloor + 1$. Then $p \geq \frac{L}{2\epsilon}$, and, in view of Theorem 1.1.1, we have $f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*) \leq \frac{L}{2p} \leq \epsilon$. Note that we need to call the oracle at p^n points. ■

Thus, $\mathscr{A}(\mathscr{G})$ justifies an *upper* complexity bound for our problem class.
But we still get some questions:

1. It may happen that our proof is too rough and the real performance of method $\mathscr{G}(p)$ is much better.
2. We still cannot be sure that $\mathscr{G}(p)$ is reasonable method for solving problem 1.2. There could exist other schemes with much higher performance.

In order to answer these questions, we need to derive the *lower* complexity bounds for problem class 1.2. The main features of such bounds are as follows.

- They are based on the *Black Box Concept* in Assumption 1.1.1.
- These bounds are valid for all reasonable iterative schemes. Thus, they provide us with a lower estimate for the analytical complexity of the problem class.
- Very often such bounds employ the idea of a *resisting oracle*.

Resisting Oracle

A *resisting oracle* tries to create the *worst possible* problem for each particular method.

- It starts from an "empty" function and it tries to answer each call of the method in the worst way.
- The answers must be *compatible* with the *previous answers* and with *description of the problem class*.
- After termination of the method, it is possible to *reconstruct* a problem which perfectly fits the final informational set accumulated by the algorithm.
- If we run the method on this newborn problem, it will reproduce the same sequence of test points since it will have the same sequence of answers from the oracle.

Example: Let's consider gradient descent on a Lipschitz-smooth, convex function f (a convex function whose gradient is Lipschitz-continuous). Gradient descent generates a sequence of points $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}}$ which satisfies

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \gamma \boldsymbol{\nabla} f(\boldsymbol{x}_k).$$

Now we are interested in the worst case analysis. We would like to find the "worst" sequence $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}}\}$ we could probably get given by gradient descent. But under the assumption that f is convex and Lipschitz-smooth, the resisting oracle here would give you "bad" values of $\boldsymbol{\nabla} f(\boldsymbol{x}_k)$, which lead to slow convergence but are still "possible" in the sense that they are values for which we can construct a convex Lipschitz-smooth function whose gradient evaluated at $\boldsymbol{x}_k$ gives the right values.

Attention: There is no actual function f here. There is only the oracle giving us values when we ask it for $\boldsymbol{\nabla} f(\boldsymbol{x}_k)$. The oracle is constructed in such a way that the values it returns could plausibly be the gradient values of some convex Lipschitz smooth function, but there is no fixed function that we start with.

Theorem 1.1.2

For $\epsilon < \frac{1}{2}L$, the analytical complexity of problem class $\mathscr{P}_\infty$ is at least $\left\lfloor \frac{L}{2\epsilon} \right\rfloor^n$ calls of oracle.

Proof of Theorem 1.1.2: Let $p = \left\lfloor \frac{L}{2\epsilon} \right\rfloor$ (≥ 1). Assume that there exists a method which needs $N < p^n$ calls of oracle to solve any problem. Let us apply this method to the following resisting oracle:

$$\text{Return } f(\boldsymbol{x}) = 0 \text{ at any test point } \boldsymbol{x}.$$

Therefore this method can find only $\bar{\boldsymbol{x}} \in B_n$ with $f(\bar{\boldsymbol{x}}) = 0$.

However, since $N < p^n$, there exists a multi index $\hat{\alpha}$ such that there were no test points in the box $X_{\hat{\alpha}}$. Define $\boldsymbol{x}_* = \boldsymbol{x}_{\hat{\alpha}}$, and consider the function

$$\bar{f}(\boldsymbol{x}) = \min\{0, L\|\boldsymbol{x} - \boldsymbol{x}_*\|_\infty - \epsilon\}.$$

Clearly, this function is l_∞-Lipschitz continuous with constant L, and its global optimal value is $-\epsilon$. Moreover, $\forall \boldsymbol{x} \notin X_{\hat{\alpha}},\ \bar{f}(\boldsymbol{x}) \neq 0$. Thus, $\bar{f}(\cdot)$ is equal to zero *at all test points* of our method (*since we assume that there is no test points in $X_{\hat{\alpha}}$.*)

Since the accuracy of the output of our method is ϵ, we come to the following conclusion: **If the number of calls of the oracle is less than p^n, then the accuracy of the result cannot be better than ϵ.**

Thus, the desired statement is proved. ■

Theorem 1.1.3

Let us compare its efficiency estimate with the lower bound:

$$\mathscr{G} : \left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor + 1 \right)^n \Leftrightarrow \textit{Lower bound: } \left\lfloor \frac{L}{2\epsilon} \right\rfloor^n$$

If $\epsilon \leq O\left(\frac{L}{n}\right)$, then the lower and upper bounds coincide up to an absolute constant multiplicative factor. This means that, for such level of accuracy, $\mathscr{G}(\cdot)$ is optimal *for the problem class $\mathscr{P}_\infty$.*

Proof of Theorem 1.1.3: Since $\epsilon \leq O\left(\frac{L}{n}\right)$, there exists $M > 0$ that satisfies

$$\epsilon \leq M \left| \frac{L}{n} \right| = M \frac{L}{n} \Leftrightarrow \frac{L}{2\epsilon} \geq \frac{n}{2M}.$$

So we can get

$$1 \leq \frac{\left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor + 1\right)^n}{\left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor\right)^n} = 1 + \sum_{k=1}^{n} \frac{c_k}{\left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor\right)^k} \leq 1 + \sum_{k=1}^{n} \frac{C_n^k}{\left(\left\lfloor \frac{n}{2M} \right\rfloor\right)^k}.$$

As $n \to \infty$, we can also get

$$\lim_{n\to\infty}\left[1+\sum_{k=1}^{n}\frac{c_k}{\left(\lfloor\frac{n}{2M}\rfloor\right)^k}\right]=1.$$

So we can get the result based on *Sandwich theorem*:

$$\lim_{n\to\infty}\frac{\left(\lfloor\frac{L}{2\epsilon}\rfloor+1\right)^n}{\left(\lfloor\frac{L}{2\epsilon}\rfloor\right)^n}=1.$$

Therefore, the statement is proofed. ■

Summerization

We aim to find the complexity bounds for global optimization. Here we use the example of *n-dimensional box problem* just as 1.2.

- In order to deal with this kind of problems, we propose the *Uniform Grid Method* $\mathscr{G}(p)$ just as Algorithm 2. The computational complexity is p^n at most. So the question is: in order to Let the accuracy get to $\epsilon > 0$, how should p be set?
- Here we assume that the objective function $f(\cdot)$ is *Lipschitz continuous* in Assumption 1.1.2. Based on the condition of Lipschitz continuous, we can get the relationship between $f(\bar{\boldsymbol{x}})$ and f^* according to Theorem 1.1.1: $f(\bar{\boldsymbol{x}}) - f^* \leq \frac{L}{2p}$.
- So, in order to satisfy $f(\bar{\boldsymbol{x}}) - f^* \leq \frac{L}{2p} \leq \epsilon$, we can set $p = \lfloor\frac{L}{2\epsilon}\rfloor + 1$. So the upper complexity is $\mathscr{A}(\mathscr{G}) = p^n = \left(\lfloor\frac{L}{2\epsilon}\rfloor + 1\right)^n$ just as Corollary 1.1.1.

1.1.4 Identity Cards of the Fields