Introductory Lectures on Optimization

Introduction

Hui Qian
qianhui@zju.edu.cn

College of Computer Science, Zhejiang University

September 11, 2024

Outline

Part I
General Formulation of The Optimization Problem

General Formulation of The Optimization Problem

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

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

and $f_0(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ be some real-valued functions defined on a set $S \subseteq \mathbb{R}^n$. We consider different variants of the following general minimization problem:

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

where the sign & can be $\leq$, $\geq$, or $=$.
*You can replace $\geq$ and $=$ with $\leq$, so $\leq$ will be used throughout the rest of the text.

General Formulation of The Optimization Problem

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

We call $f_0(\boldsymbol{x})$ the objective function of our problem, the vector function

$$f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^\top$$

is called the vector of functional constraints, the set S is the basic feasible set, and the set

$$Q = \{\boldsymbol{x} \in S \mid f_j(\boldsymbol{x}) \le 0, j = 1 \ldots m\}$$

is the (entire) feasible set of problem (1). It is just a convention to consider minimization problems.

Natural Classification

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

There exists a natural classification of the types of minimization problems:

- constrained problems: $Q \subset \mathbb{R}^n$.
- unconstrained problems: $Q \equiv \mathbb{R}^n$.
- smooth problems: all $f_j(\boldsymbol{x})$ are differentiable.
- nonsmooth problems: there are several non-differentiable components $f_k(\boldsymbol{x})$.

Natural Classification

$$\boxed{\begin{aligned} &\min f_0(\boldsymbol{x}) \\ \text{s.t.}\quad & f_j(\boldsymbol{x}) \ \leq\ 0, j = 1 \dots m, \\ & \boldsymbol{x} \in S. \end{aligned} \qquad (1)}$$

- linearly constrained problems: the functional constraints are affine like

$$f_j(\boldsymbol{x}) = \sum_{i=1}^{n} a_j^{(i)} \boldsymbol{x}^{(i)} + b_j \equiv \langle a_j,\ \boldsymbol{x}\rangle + b_j, \quad j = 1, \dots m.$$

  - linear optimization problem: if $f_0(\cdot)$ is also affine.
  - quadratic optimization problem: if $f_0(\cdot)$ is quadratic.
  - quadratically constrained quadratic problem: if all the functions $f_0(\cdot), \cdot, f_m(\cdot)$ are quadratic.

Classification based on the properties of feasible set

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

$$Q = \{\boldsymbol{x} \in S \mid f_j(x) \le 0, j = 1 \dots m\}$$

There is also a classification based on properties of the feasible set.

- Problem (1) is called feasible, if $Q \neq \emptyset$。
- Problem (1) is called strictly feasible, if there exists an $\boldsymbol{x} \in \operatorname{int} Q$ such that $f_j(\boldsymbol{x}) < 0$ (or > 0) for all inequality constraints and $f_j(\boldsymbol{x}) = 0$ for all equality constraints (Slater condition).

Different Types of Solutions

$$\boxed{\begin{aligned} & \min f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & f_j(\boldsymbol{x}) \;\le\; 0, j = 1 \ldots m, \\ & \boldsymbol{x} \in S, \end{aligned} \qquad (1)}$$

Finally, we distinguish different types of solutions to (1):

- $\boldsymbol{x}^*$ is called the global optimal solution to (1) if $f_0(\boldsymbol{x}^*) \le f_0(\boldsymbol{x})$ for all $\boldsymbol{x} \in Q$. In this case, $f_0(\boldsymbol{x}^*)$ is called the (global) optimal value of the problem. ($\boldsymbol{x}^* = \operatorname{argmin} \cdots$)
- $\boldsymbol{x}^*$ is called a local optimal solution to (1) if $\exists\delta$, for all $\boldsymbol{x} \in \text{NBR}\,(\boldsymbol{x}^*, \delta) \cap Q$, we have

$$f_0(\boldsymbol{x}^*) \le f_0(\boldsymbol{x}).$$

Examples Demonstrating The Origin of The Problem

Example 1 (Example 1.1.1 of Nesterov [2003])

Let $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ be our design variables. Then we can fix some functional characteristics of our decision vector $\boldsymbol{x}$: $f_0(\boldsymbol{x})$, ..., $f_m(\boldsymbol{x})$. For example,we can consider a price of the project, amount of required resources, reliability of the system, etc. We fix the most important characteristic, $f_0(\boldsymbol{x})$, as our objective. For all others, we impose some bounds: $a_j \leq f_j(\boldsymbol{x}) \leq b_j$. Thus, we come to the problem:

$$\begin{aligned} & \min f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & a_j \leq f_j(\boldsymbol{x}) \leq b_j, j = 1 \ldots m, \\ & \boldsymbol{x} \in S, \end{aligned}$$

where S stands for the structural, constraints like non-negativity, boundedness of some variables, etc.

Examples Demonstrating The Origin of The Problem

Example 2 (Example 1.1.2 of Nesterov [2003])

Let our initial problem be as follows:

$$\text{Find} \quad \boldsymbol{x} \in \mathbb{R}^n \quad \text{such that} \quad f_j(\boldsymbol{x}) = a_j, \quad j = 1 \ldots m. \tag{2}$$

Then we can consider the problem

$$\min_{\boldsymbol{x}} \sum_{j=1}^{m} (f_j(\boldsymbol{x}) - a_j)^2,$$

perhaps even with some additional constraints on $\boldsymbol{x}$. If the optimal value of the latter problem is zero, we conclude that our initial problem (2) has a solution.

Examples Demonstrating The Origin of The Problem

Example 3 (Example 1.1.3 of Nesterov [2003])

Sometimes our decision variables $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ must be integers. This can be described by the following constraint:

$$\sin(\pi \boldsymbol{x}^{(i)}) = 0, \; i = 1 \ldots n.$$

Thus, we can also treat integer optimization problems:

$$\begin{aligned} & \min f_0(\boldsymbol{x}) \\ \text{s.t.:} \quad & a_j \leq f_j(\boldsymbol{x}) \leq b_j, \; j = 1 \ldots m, \\ & \boldsymbol{x} \in S, \\ & \sin(\pi \boldsymbol{x}^{(i)}) = 0, \; i = 1 \ldots n. \end{aligned}$$

Optimism or Pessimism

The optimism of the pioneers of Nonlinear Optimization (Nesterov [2003]):

> Nonlinear Optimization is a very important and promising application theory. It covers almost ALL needs of Operations Research and Numerical Analysis. (AI and Machine Learning ?)

The pessimism of realist:

> In general, optimization problems should be UNSOLVABLE (?)

Indeed, from our real-life experience, it is difficult to believe in the existence of a universal tool which is able to solve all problems in the world.

Part II
Interplay of Optimization and Machine Learning

The Machine Learning (Supervised) Paradigm

Consider the following spaces: a space of *examples* $\mathcal{X}$, a space of labels $\mathcal{Y}$, and a space of *hypothesis* $\mathcal{H}$ that contains functions mapping $\mathcal{X}$ to $\mathcal{Y}$. Also consider a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and a probability measure P over the space $\mathcal{X} \times \mathcal{Y}$.

Definition 4

For any hypothesis $h \in \mathcal{H}$ we define the risk of h with respect to L and P to be:

$$\mathcal{L}_P(h) = \mathrm{E}_P\left(L\left(h(\boldsymbol{X}), Y\right)\right), \quad \text{with } (\boldsymbol{X}, Y) \sim P.$$

The general problem of machine learning can be then cast as finding the hypothesis $h \in \mathcal{H}$ that solves the following optimization problems:

$$\min_{h \in \mathcal{H}} \mathcal{L}_P(h).$$

*We here follow the narrative in [Munoz, 2014]

Empirical Risk Minimization

The practical way is to solve an *empirical risk minimization* problem. That is

$$\min_{h\in\mathcal{H}}\sum_{i=1}^{m}L\left(h(\boldsymbol{x}_i),y_i\right)=\min_{\theta}\sum_{i=1}^{m}L\left(h_\theta(\boldsymbol{x}_i),y_i\right),$$

since we do not know the P exactly. That is

$$\theta_*=\operatorname*{argmin}_{\theta}\sum_{i=1}^{m}L\left(h_\theta(\boldsymbol{x}_i),y_i\right).$$

1. Classification: the output variable takes class labels. Especially, for binary classification, $y_i\in\{1,-1\}$.
2. Regression: the output variable takes continuous values. That is $y_i\in\mathbb{R}$.

Classification: The 0∼1 Loss

By using the 0∼1 *loss*, we can rewrite the objective function as:

$$\sum_{i=1}^{m} \mathbf{1}_{\text{sign}(h(\boldsymbol{x}_i)) \neq y_i} = \sum_{i=1}^{m} \mathbf{1}_{y_i h(\boldsymbol{x}_i) < 0}$$

where $\mathbf{1}_z$ is just the indicator function (Heaviside step function, or the unit step function), that is

$$\mathbf{1}_z = \begin{cases} 1, & z \text{ is true}, \\ 0, & \text{otherwise}. \end{cases}$$

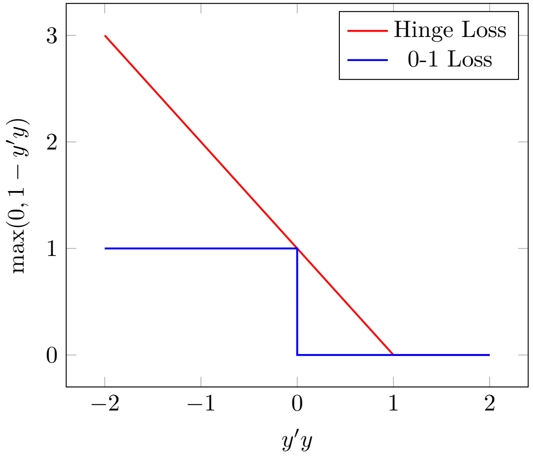
In fact, this problem is NP-hard so it is pointless to try to minimize this function.

Classification: The Surrogate of The 0∼1 Loss

To deal with this problem, we can introduce some *surrogate* like the *hinge loss* or *margin loss* given by $\phi_h(y, y') = \max(0, 1 - y'y)$.

Remark. This function is an upper bound to the 0∼1 loss as shown in the right figure. Coarsely speaking, it is much more easy to solve (We will explain it later). And, it turns out [Bartlett et al., 2006] that

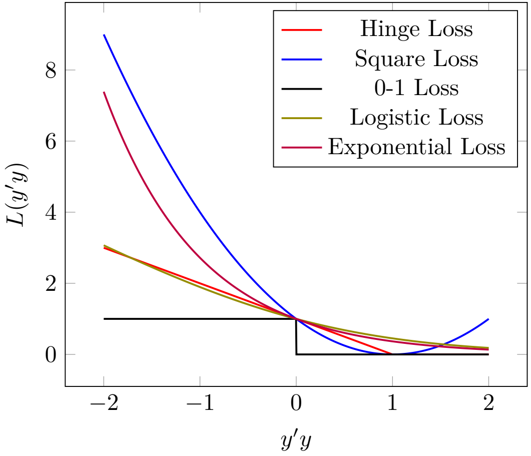
$$\min_{h\in\mathcal{H}} \mathrm{E}\left[L\left(h(\boldsymbol{x}), y\right)\right] = \min_{h\in\mathcal{H}} \mathrm{E}\left[\phi_h\left(h(\boldsymbol{x}), y\right)\right].$$



Classification: The Surrogate of The 0∼1 Loss

Other common loss functions for classification are list as follows.

1. Square loss:
   $\phi_s(y, y') = (1 - yy')^2.$
2. Logistic loss:
   $\phi_l(y, y') = \frac{1}{\ln 2} \ln(1 + e^{-yy'}).$
3. Exponential loss:
   $\phi_e(y, y') = e^{(-yy')}.$
4. Cross entropy loss:
   $\phi_c(y, y') = -t \ln(y') - (1 - t) \ln(1 - y')$
   where $t = (1 + y)/2.$



Maximum Likelihood Estimation View

Let us say, we have a likelihood function $P(\boldsymbol{X}, Y|\theta)$. Then the MLE for θ, the parameter we want to infer, is

$$\begin{aligned}\theta_{\text{MLE}} &= \operatorname*{argmax}_{\theta} P(\boldsymbol{X}, Y|\theta) = \operatorname*{argmax}_{\theta} \log P(\boldsymbol{X}, Y|\theta)\\ &= \operatorname*{argmax}_{\theta} \log \prod_i P(\boldsymbol{x}_i, y_i|\theta)\\ &= \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log P(\boldsymbol{x}_i, y_i|\theta).\end{aligned}$$

For case that $P(\boldsymbol{x}_i, y_i|\theta) \sim \exp -L(h_\theta(\boldsymbol{x}_i), y_i)$, we have

$$\theta_{\text{MLE}} = \operatorname*{argmin}_{\theta} \sum_{i=1}^{n} L(h_\theta(\boldsymbol{x}_i), y_i).$$

Maximum A Posteriori Estimation View

If we replace the likelihood in the MLE formula above with the posterior, we get:

$$\begin{aligned}\theta_{\text{MAP}} &= \operatorname*{argmax}_{\theta} P(\boldsymbol{X}, Y|\theta)P(\theta) = \operatorname*{argmax}_{\theta} \log\{P(\theta)P(\boldsymbol{X}, Y|\theta)\} \\ &= \operatorname*{argmax}_{\theta} \log\left\{P(\theta)\prod_i P(\boldsymbol{x}_i, y_i|\theta)\right\} \\ &= \operatorname*{argmax}_{\theta} \left\{\log P(\theta) + \sum_{i=1}^{n} \log P(\boldsymbol{x}_i, y_i|\theta)\right\}.\end{aligned}$$

For case that $P(\boldsymbol{x}_i, y_i|\theta) \sim \exp -L(h_\theta(\boldsymbol{x}_i), y_i)$ and $P(\theta) \sim \exp -\lambda \|\theta\|_2^2$, we have

$$\theta_{\text{MAP}} = \operatorname*{argmin}_{\theta} \left\{\sum_{i=1}^{n} L(h_\theta(\boldsymbol{x}_i), y_i) + \lambda \|\theta\|_2^2\right\}.$$

Some Remarks

We remark here:

1. Hypothesis: Not only shallow model, deep learning ...
2. Variational Inference: Minimize KL divergence.
3. Reinforcement Learning: Maximize a long-term reward.
4. Optimization vs. Sampling.

Part III
Performance of Numerical Methods

Method and Class of Problem

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

Consider a method for solving problem (1): which does nothing except report that $x^* = 0$。

- For those which have the optimal solution exactly at the origin, the "performance" of this method is unbeatable.
- Also, this method does not work properly for any other problems.

Hence, we cannot speak about the best method for a particular problem P, but we can do so for a class of problems $\mathcal{F} \ni P$.

The performance of a method $\mathcal{M}$ on the whole class $\mathcal{P}$ can be a natural measure of its efficiency

Method and Class of Problem

Consider the performance of $\mathcal{M}$ on a class $\mathcal{P}$. We should assume that the method $\mathcal{M}$ does not have complete information about a particular problem $\mathcal{P}$.

- Model:
  The known (to a numerical scheme) "part"of problem $\mathcal{P}$ is called the model of the problem. We denote the model by Σ. Usually the model consists of the formulation of the problem, description of classes of functional components, etc.
- Oracle:
  We describe the process of collecting the specific information about $\mathcal{P}$ via the notion of an oracle. An oracle $\mathcal{O}$ is just a unit which answers the successive questions of the methods. The method $\mathcal{M}$ is trying to solve the problem $\mathcal{P}$ by collecting and handling the answers.

In general, each problem can be described by different models. Moreover, for each problem we can develop different types of oracles.

Method and Class of Problem

Let us fix Σ and $\mathcal{O}$. In this case, it is natural to define the performance of $\mathcal{M}$ on $(\Sigma, \mathcal{O})$. as its performance on the worst P_w from $(\Sigma, \mathcal{O})$. Note that this P_w can be bad only for $\mathcal{M}$.

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

what does "to solve the problem"mean?

> Solving the problem means finding an approximate solution to $\mathcal{P}$ with some accuracy $\epsilon > 0$.

General Iterative Scheme

General Iterative Scheme

- Input: Starting point $\boldsymbol{x}_0$ and accuracy $\epsilon > 0$
- Initialization: Set $k = 0$, $I_{-1} = \emptyset$

Main Loop:

```
1  Call oracle $\mathcal{O}$ at point $\boldsymbol{x}_k$.
2  Update the informational set: $I_k = I_{k-1} \cup (\boldsymbol{x}_k, \mathcal{O}(\boldsymbol{x}_k))$。
3  Apply the rules of method $\mathcal{M}$ to $I_k$ and generate a new point $\boldsymbol{x}_{k+1}$.
4  Check criterion $\mathcal{T}_\epsilon$. If yes then form an output $\bar{\boldsymbol{x}}$. Otherwise set $k := k+1$ and go to Step 1.
```

(3)

Complexity

In the scheme (3), we can see two potentially expensive steps. The first one is Step 1, where we call the oracle. The second one is Step 3, where we form the new test point. Thus, we can introduce two measures of complexity of problem $\mathcal{P}$ for method $\mathcal{M}$:

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

For a particular method $\mathcal{M}$ as applied to problem $\mathcal{P}$, arithmetical complexity can be easily obtained from the analytical complexity and complexity of the oracle.

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 the Local Black Box Concept , is listed in the following lines:

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.

Oracle

The standard formulation (1) is called a functional model of optimization problems. Usually, for such models the standard assumptions are related to the level of smoothness of functional components. According to the degree of smoothness we can apply different types of oracle:

- zeroth-order oracle:
  return $f(\boldsymbol{x})$.
- first-order oracle:
  return $f(\boldsymbol{x})$and gradient $\nabla f(\boldsymbol{x})$.
- second-order oracle:
  return $f(\boldsymbol{x})$, gradient $\nabla f(\boldsymbol{x})$ and Hessian $\nabla^2 f(\boldsymbol{x})$.

Part IV
Complexity Bounds for Global Optimization

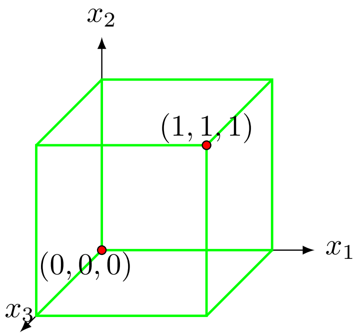
N-dimensinal Box Constraint Problem

Consider the following problem:

$$\min_{\boldsymbol{x} \in \mathbb{B}_n} f(\boldsymbol{x}). \tag{4}$$

In our terminology, this is a constrained minimization problem with no functional constraints. The basic feasible set of this problem is $\mathbb{B}_n$, an n-dimensional box in $\mathbb{R}^n$:

$$\mathbb{B}_n = \{\boldsymbol{x} \in \mathbb{R}^n | 0 \le \boldsymbol{x}^{(i)} \le 1, i = 1 \dots n\}.$$



N-dimensinal Box Constraint Problem

Let us measure distances in $\mathbb{R}^n$ by the ℓ_∞:

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

Assume that, with respect to this norm, the objective function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous on $\mathbb{B}_n$:

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le L \|\boldsymbol{x} - \boldsymbol{y}\|_\infty, \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{B}_n, \tag{5}$$

with some constant L (Lipschitz constant).

Uniform Grid Method

Method $\mathcal{G}(p)$

1. Form $(p+1)^n$ points

$$\boldsymbol{x}_{(i_1,\dots,i_n)} = \left(\frac{i_1}{p}, \frac{i_2}{p}, \cdots, \frac{i_n}{p}\right)^\top, \tag{6}$$

where $(i_1, \dots, i_n) \in \{0, \dots, p\}^n$. (i.e., $i_k \in \{0, \dots, p\}$ for $k = 1 \dots n$.)

2. Among all points $\boldsymbol{x}_{(i_1,\dots,i_n)}$, find the point $\bar{\boldsymbol{x}}$ with the minimal value of the objective function.

3. The pair $(\bar{\boldsymbol{x}}, f(\bar{\boldsymbol{x}}))$ is the output of the method.

Uniform Grid Method

This method forms a uniform grid of the test points inside the box $\mathbb{B}_n$, computes the best value of the objective over this grid, and returns this value as an approximate solution to problem (4).

- Zero-order iterative method,
- Without any influence from the accumulated information on the sequence of test points.

Theorem 5 (Theorem 1.1.1 of Nesterov [2003])

Let f^* be a global optimal value of problem (4). Then

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

Uniform Grid Method

Proof.

Let $\boldsymbol{x}_*$ be a global minimum of our problem. Then there exist coordinates $(i_1, i_2, \ldots, i_n)$ such that

$$\boldsymbol{x} \equiv \boldsymbol{x}_{(i_1, i_2, \ldots, i_n)} \leq \boldsymbol{x}_* \leq \boldsymbol{x}_{(i_1+1, i_2+1, \ldots, i_n+1)} \equiv \boldsymbol{y}.$$

(here and in the sequel we write $\boldsymbol{x} \leq \boldsymbol{y}$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ if and only if $\boldsymbol{x}^{(i)} \leq \boldsymbol{y}^{(i)}$ for all $i = 1 \ldots n$.) Note that

$$\boldsymbol{y}^{(i)} - \boldsymbol{x}^{(i)} = \frac{1}{p},$$

for $i = 1 \ldots n$, and

$$\boldsymbol{x}_*^{(i)} \in [\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}], \quad i = 1 \ldots n.$$

Uniform Grid Method

Proof. (continued)

Denote $\hat{\boldsymbol{x}} = (\boldsymbol{x}+\boldsymbol{y})/2$. Let us form a point $\tilde{\boldsymbol{x}}$ as follows:

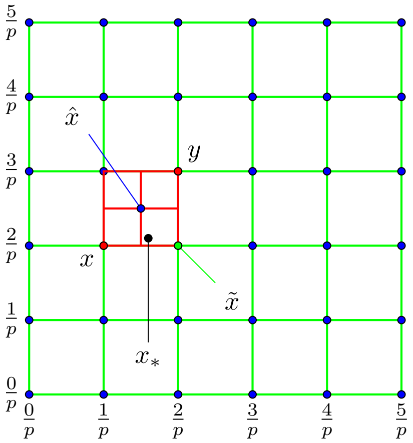
$$\tilde{\boldsymbol{x}} = \begin{cases} \boldsymbol{y}^{(i)}, & \text{if } \boldsymbol{x}_*^{(i)} \geq \hat{\boldsymbol{x}}^{(i)}, \\ \boldsymbol{x}^{(i)}, & \text{otherwise.} \end{cases}$$

It is clear that $|\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}_*^{(i)}| \leq \frac{1}{2p}$, $i = 1 \dots n$. Therefore

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}_*\|_\infty = \max_{1 \leq i \leq n} |\tilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}_*^{(i)}| \leq \frac{1}{2p}.$$

Since $\tilde{\boldsymbol{x}}$ belongs to our grid, we conclude that

$$f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}_*) \leq f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x}_*) \leq L \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_*\|_\infty \leq \frac{L}{2p}. \quad \square$$


$\frac{0}{p}$ $\frac{1}{p}$ $\frac{2}{p}$ $\frac{3}{p}$ $\frac{4}{p}$ $\frac{5}{p}$
$\hat{x}$
y
x
$\tilde{x}$
x_*

$p = 5, n = 2, N = (5+1)^2 = 36$

Upper Complexity Bound

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

$$\text{Find } \bar{\boldsymbol{x}} \in \mathbb{B}_n : f(\bar{\boldsymbol{x}}) - f^* \leq \epsilon. \tag{7}$$

Then we immediately get the following result.

Corollary 6 (Corollary 1.1.1 of Nesterov [2003])

The analytical complexity of problem class (4)、(5)、(7) for method $\mathcal{G}$ is at most

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

(here and in the sequel, $\lfloor a \rfloor$ is the integer part of $a \in \mathbb{R}$).

Upper Complexity Bound

Proof.

Take $p = \lfloor \frac{L}{2\epsilon} \rfloor + 1$. Then $p \geq \frac{L}{2\epsilon}$, and, in view of Theorem 5, we have

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

Note that we construct $(p+1)^n$ points. □

Remark.

- Each point has an access to oracle, so the number of accessing is equal to the number of points.
- If $\frac{L}{2p} \leq \epsilon$, we have $p \geq \frac{L}{2\epsilon}$. Thus, we set $p = \lfloor \frac{L}{2\epsilon} \rfloor + 1$ and get $(p+1)^2$ points.
- Thus, $\mathcal{A}(\mathcal{G})$ justifies an upper complexity bound for our problem class.

Lower Complexity Bound

Two situations should be further explored:

- Firstly,it may happen that our proof is too rough and the real performance of method $\mathcal{G}(p)$ is much better.
- Secondly, we still cannot be sure that $\mathcal{G}(p)$ is a reasonable method for solving (4). There could exist other schemes with much higher performance.

In order to answer these questions, we need to derive lower complexity bounds for the problem class (4)、(5)、(7).The main features of such bounds are as follows.

- They are based on the Black Box Concept.
- 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.

Lower Complexity Bound

A resisting oracle tries to create the worst possible problem for each particular method (for example, $\mathcal{G}(p)$).

- It starts from an "empty"function and it tries to answer each call of the method in the worst possible way.
- However, 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. Moreover, 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.

Lower Complexity Bound

Let us show how this works for problem (4). Consider the class of problems $\mathcal{C}$ defined as follows.

Model:	$\min_{\boldsymbol{x} \in \mathbb{B}_n} f(\boldsymbol{x})$, where $f(\boldsymbol{x})$ is ℓ_∞-Lipschitz continuous on $\mathbb{B}_n$.
Oracle:	Zero-order Local Black Box.
Approximate solution:	Find $\bar{\boldsymbol{x}} \in \mathbb{B}_n : f(\bar{\boldsymbol{x}}) - f^* \leq \epsilon$.

Theorem 7 (Theorem 1.1.2 of Nesterov [2003])

For $\epsilon < \frac{1}{2}L$, the analytical complexity of problem class $\mathcal{C}$ is at least $\left(\lfloor \frac{L}{2\epsilon} \rfloor\right)^n$.

Approach to the proof: There always exists a scenario (worse problem) where, regardless of how the sampling is done, when the sample number $N < \left(\lfloor \frac{L}{2\epsilon} \rfloor\right)^n$, the error is equal to ϵ.

Lower Complexity Bound

Proof.

Denote $q = \lfloor \frac{L}{2\epsilon} \rfloor (\geq 1)$. Assume that there exists a method, which need $N < q^n$ calls of oracle to solve any problem from $\mathcal{C}$. Let us apply this method to the following resisting strategy:

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

Therefore this method can find only $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ with $f(\bar{\boldsymbol{x}}) = 0$. However, note that there exists $\hat{\boldsymbol{x}} \in \mathbb{B}_n$ such that

$$\hat{\boldsymbol{x}} + \frac{1}{q}\boldsymbol{e} \in \mathbb{B}_n, \quad \boldsymbol{e} = (1, \ldots, 1)^\top \in \mathbb{R}^n,$$

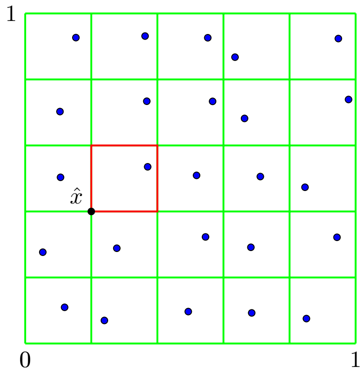
and there were no test points inside (not on the boundary of) the box $\mathbb{B} = \{\boldsymbol{x} | \hat{\boldsymbol{x}} \leq \boldsymbol{x} \leq \hat{\boldsymbol{x}} + \frac{1}{q}\boldsymbol{e}\}$.

Lower Complexity Bound

Remark.
To prove the existence of an $\hat{x}$ that can construct a $\mathbb{B}$ without any test points, it suffices to show:

The statement 'Every $\mathbb{B}$ constructed by every $\hat{x}$ contains at least one test point inside' is false (in conflict with the assumption).

The proof method involves providing a counterexample.



$q = 5, n = 2, N = 25$

Lower Complexity Bound

Proof. continued

Denotes $\boldsymbol{x}_* = \hat{\boldsymbol{x}} + \frac{1}{2q}\boldsymbol{e}$. Consider the function

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

Clearly, (1) this function is ℓ_∞-Lipschitz continuous with the constant L and (2) its global optimal value is $-\epsilon$. Moreover, $\bar{f}(x)$ differs from zero only inside the box $\mathbb{B}' = \{\boldsymbol{x}|\ \|\boldsymbol{x} - \boldsymbol{x}_*\|_\infty \leq \frac{\epsilon}{L}\}$. Since $2q \leq L/\epsilon$, we conclude that

$$\mathbb{B}' \subseteq \mathbb{B} \equiv \{\boldsymbol{x}|\ \|\boldsymbol{x} - \boldsymbol{x}_*\|_\infty \leq \frac{1}{2q}\}.$$

Thus, $\bar{f}(\boldsymbol{x})$ is equle to zero at all test points of our methods. Since the accuracy of the rest of our method is ϵ, we come to the following conclusion: If the number of calls of the oracle is less that q^n, then the accuracy of the result cannot be better that ϵ.

Lower Complexity Bound

Now we can say much more about the performance of the Uniform Grid Method.Let us compare its efficiency estimate with the lower bound:

$$\mathcal{G} : \left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor + 2 \right)^n, \text{ Lower Bound: } \left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor \right)^n.$$

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, $\mathcal{G}(p)$ is optimal for the problem class $\mathcal{C}$.

Meanwhile, the theorem 7 supports our original statement that general optimization problems are unsolvable.

Discussion for Lower Complexity Bound

Remark.
If $\epsilon \leq O\left(\frac{L}{n}\right)$, we have

$$\frac{\left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor + 2\right)^n}{\left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor\right)^n} = \left(1 + \frac{1}{\frac{1}{2}\left\lfloor \frac{L}{2\epsilon} \right\rfloor}\right)^n \leq \left(1 + \frac{1}{cn}\right)^n = \left(\left(1 + \frac{1}{cn}\right)^{cn}\right)^{\frac{1}{c}} \underset{n \to \infty}{=} (e)^{\frac{1}{c}},$$

where c is a constant, since

$$\lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n = e.$$

Discussion for Lower Complexity Bound

Example 8

Consider the problem class $\mathcal{F}$: $L = 2,\ n = 10,\ \epsilon = 0.01$. Note that the size of the problem is very small and we ask only for 1% accuracy. The lower complexity bound for this class is $\left(\frac{L}{2\epsilon}\right)^n$. Let us compute it for our examples.

Lower bound:	10^{20} calls of the oracle
Oracle complexity:	at least n arithmetic operations, (a.o.)
Total complexity:	10^{21} a.o.
Processor performance	10^6 a.o. per second
Total time:	10^{15}s
One year	less than $3.2 \cdot 10^7$s
We need:	$31,250,000$ 年

Discussion for Lower Complexity Bound

This estimate is so disappointing that we cannot maintain any hope that such problems may become solvable in the future. Let us just play with the parameters of the problem class.

- If we change n to $n+1$, then the estimate is multiplied by one hundred. Thus, for $n = 11$ our lower bound is valid for a much more powerful computer.
- On the contrary, if we multiply ϵ by two, we reduce the complexity by a factor of a thousand. For example, if $\epsilon = 8\%$, then we need only two weeks.

References I

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Aharon Ben-Tal and Arkadiaei Semenovich Nemirovskiaei. Lectures on modern convex optimization: analysis, algorithms, and engineering applications. 2001.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Andres Munoz. Machine learning and optimization. *URL: https://www. cims. nyu. edu/~ munoz/files/ml_optimization. pdf [accessed 2016-03-02][WebCite Cache ID 6fiLfZvnG]*, 2014.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2003.

Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

References II

Stephen J Wright and Benjamin Recht. *Optimization for data analysis*. Cambridge University Press, 2022.

Appendix: Differentiable

A function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at a point $\mathbf{a} \in \mathbb{R}^n$ if there exists a vector

$$\nabla f(\mathbf{a}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{a}), \frac{\partial f}{\partial x_2}(\mathbf{a}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{a}) \right)$$

(called the gradient) such that:

$$\lim_{\mathbf{h} \to \mathbf{0}} \frac{\|f(\mathbf{a}+\mathbf{h}) - f(\mathbf{a}) - \nabla f(\mathbf{a}) \cdot \mathbf{h}\|}{\|\mathbf{h}\|} = 0.$$

Here, $\mathbf{h}$ is a small vector in $\mathbb{R}^n$, and $\nabla f(\mathbf{a}) \cdot \mathbf{h}$ represents the dot product, which approximates the change in f when moving from $\mathbf{a}$ to $\mathbf{a} + \mathbf{h}$.

Appendix: Differentiable

Intuitions:

- **Gradient**: The gradient vector $\nabla f(\mathbf{a})$ represents the direction and rate of the steepest increase of the function f at the point $\mathbf{a}$.
- **Linear Approximation**: Differentiability means that f can be well approximated by a linear function (specifically, the tangent hyperplane defined by the gradient) near the point $\mathbf{a}$.
- **Continuity of the Gradient**: Differentiability implies that f is locally linear at the point and that the partial derivatives (components of the gradient) exist and are continuous around $\mathbf{a}$.

In summary, a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at a point if it can be closely approximated by a linear function at that point, characterized by the gradient vector.

Appendix: Differentiable

In general, if a function is **not differentiable** at a point, it means that the **gradient** at that point is not defined. Specifically, non-differentiability implies that the function cannot be approximated linearly at that point by a gradient. Therefore, the gradient either does not exist or cannot be defined at that point.
Reasons for Non-Differentiability:

- **Discontinuity**: If a function is not continuous at a point, it cannot be differentiable at that point. In this case, the gradient cannot be defined.
- **Cusp or Corner**: If the function has a cusp (e.g., the absolute value function $f(x) = |x|$ at $x = 0$) or a corner, where the left-hand and right-hand derivatives do not match, the gradient cannot be uniquely defined.
- **Vertical Tangent**: When the tangent to the function at a point is vertical, resulting in an infinite slope, for example, $f(x) = \sqrt[3]{x}$ at $x = 0$, a finite gradient cannot be defined.

In summary, if a function is not differentiable at a point, it indeed means that the gradient cannot be defined there. Non-differentiability generally implies that there is no linear function that can accurately approximate the function's behavior at that point, thus the gradient does not exist or cannot be determined.

Appendix: The Hard Margin SVM

Consider a hyper-plane for binary classification:

$$\boldsymbol{w}^\top \boldsymbol{x} + b = 0.$$

The distance from a point $\boldsymbol{x}$ to the plane $(\boldsymbol{w}, b)$ is

$$r = \frac{|\boldsymbol{w}^\top \boldsymbol{x} + b|}{\|\boldsymbol{w}\|}.$$

If $(\boldsymbol{w}, b)$ can correctly classify the training samples, that is, for $(x_i, y_i) \in D$, if $y_i = +1$, then $\boldsymbol{w}^\top \boldsymbol{x}_i + b > 0$; if $y_i = -1$, then there is $\boldsymbol{w}^\top \boldsymbol{x}_i + b < 0$. In this case, we can adjust $(\boldsymbol{w}, b)$ such that

$$\begin{cases} \boldsymbol{w}^\top \boldsymbol{x}_i + b \geq +1, & y_i = +1, \\ \boldsymbol{w}^\top \boldsymbol{x}_i + b \leq -1, & y_i = -1. \end{cases}$$

Appendix: The Hard Margin SVM

Therefore, the sum of the distances from the support vectors of different classes to the hyper-plane is:

$$\gamma = \frac{2}{\|\boldsymbol{w}\|},$$

which is called the margin of the two classes. And the optimization form of maximum margin is

$$\begin{aligned} \max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|} &\Leftrightarrow \min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|_2 \\ \text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) &\geq 1, \ i = 1 \ldots m. \end{aligned}$$

This is the hard margin SVM.

Appendix: The Hard Margin SVM

The dual problem is obtained using the Lagrange multiplier method:

$$\max_{\alpha} \min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \alpha) \triangleq \frac{1}{2}\|\boldsymbol{w}\|_2 + \sum_{i=1}^{m} \alpha_i(1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)), \tag{8}$$

where $\alpha = (\alpha_1; \alpha_2; \ldots; \alpha_m)$. To calculate the internal minimization, you can directly use the partial derivative of $\boldsymbol{w}, b$ to be 0 to find the solution:

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i, \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{9}$$

Appendix: The Hard Margin SVM

Substitute result of (9) into (8), and finally we get:

$$\begin{aligned}
&\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j, \\
\text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0 \\
& \alpha_i \geq 0, \ i = 1 \ldots m.
\end{aligned}$$

This is the dual form of the hard margin SVM. Thus, $f(x)$ can be reformulated as follows.

$$\begin{aligned}
f(\boldsymbol{x}) &= \boldsymbol{w}^\top \boldsymbol{x} + b \\
&= \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{x} + b \quad \leftarrow \text{(Using kernel trick } \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)\text{)}.
\end{aligned}$$

Appendix: The Hard Margin SVM

From the KKT conditions, we have

$$\begin{cases} \alpha_i \geq 0, \\ y_i f(\boldsymbol{x}_i) - 1 \geq 0, \\ \alpha_i (y_i f(\boldsymbol{x}_i) - 1) = 0. \end{cases}$$

Therefore, in $f(x)$, only the samples whose α_i is not 0 ($y_i f(\boldsymbol{x}_i) = 1$) are needed in the computation. We call these samples the support vectors.

Remark. SVM can be trained by SMO (Sequential Minimal Optimization) algorithm [Platt, 1998].

Appendix: The Soft Margin SVM

For the linearly inseparable case, we want to maximize the margin with as few misclassified samples as possible. Therefore, our optimization fomulation is written as:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2 + C\sum_{i=1}^{m} \max(0, 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)).$$

Introduce the slack variable ζ_i, for $i = 1, \ldots, m$, we get

$$\begin{aligned} &\min_{\boldsymbol{w}} C\sum_{i=1}^{n} \zeta_i + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & y_i(\boldsymbol{x}_i^\top \boldsymbol{w} + b) + \zeta_i \geq 1, \quad \zeta_i \geq 0. \end{aligned}$$

This is the soft margin SVM.

Appendix: The Soft Margin SVM

Just like the hard margin SVM, we have the dual form of the soft margin SVM:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j,$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0, \; i = 1\ldots m.$$

From the KKT conditions, we have

$$\begin{cases} y_i f(\boldsymbol{x}_i) - 1 + \zeta_i \geq 0, \\ \alpha_i(y_i f(\boldsymbol{x}_i) - 1 + \zeta_i) = 0, \\ \zeta_i \geq 0, \mu\zeta_i = 0, \alpha_i \geq 0. \end{cases} \quad \Rightarrow \text{Support Vectors: } y_i f(\boldsymbol{x}_i) = 1 - \zeta_i.$$

Thank You!

Email:qianhui@zju.edu.cn