Introduction to Optimization

Parin Chaipunya

[1] Department of Mathematics / [2] Center of Excellence in Theoretical and Computational Science
King Mongkut's University of Technology Thonburi

Multi-agent optimization
Mathematical economics
Alexandrov geometry in optimization
Optimal transport

Topic Overview

This introduction shall covers the following topics:

- Introduction to Optimization Modeling
- Unconstrained Optimization
  - Principles of Unconstrained Optimization
  - Gradient Descent Algorithm and its Variants
  - Numerical Examples
- Constrained Optimization
  - Principles of Constrained Optimization
  - Constrained Optimization Solvers
- Heuristic Approach

Introduction to Optimization

What is Optimization?

The term **OPTIMIZATION** is almost self-explained:

“We are interested in choosing the best choice out of the vastly available options.”

Best choice that produces

- Maximum profit
- Cheapest cost
- Fastest route
- Maximum likelihood
- Least error
- etc.

Ingredients

What do we need in an optimization problem?

- Decision variables: $x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$.
- Objective function: $f : \mathbb{R}^n \to \mathbb{R}$ to be minimized/maximized.
- Constraint: A subset $C \subseteq \mathbb{R}^n$.

A general formulation of an optimization is

$$\begin{cases} \min / \max & f(x) \\ \text{s.t.} & x \in C. \end{cases}$$

Note that miximizing $f(x)$ over C is equivalent to minimizing $-f(x)$ over C. Hence the optimization theory is developed focusing on minimization.

Types of solutions.

$$\begin{cases} \min & f(x) \\ \text{s.t.} & x \in C. \end{cases}$$

- **Global solution [$\bar{x} \in Min(f, C)$] :** $f(\bar{x}) \leq f(x)$ for all $x \in C$.
- **Local solution [$\bar{x} \in LMin(f, C)$] :** $f(\bar{x}) \leq f(x)$ for all $x \in C$ that is *near* $\bar{x}$.

If $C = \mathbb{R}^n$, then there is no constraint, i.e. an Unconstrained optimization. In this case, we adopt shorthand notations $Min(f)$ and $LMin(f)$ for the above notions.

Situations

$$\begin{cases} \max & \text{Profit} \\ \text{s.t.} & \text{Budget constraints} \\ & \text{Resource constraints} \\ & \text{Market rules} \end{cases}$$

Situations

$$\begin{cases} \min & \text{Production cost} \\ \text{s.t.} & \text{Demand constraints} \\ & \text{Resource constraints} \\ & \text{Infrastructure constriants} \end{cases}$$

Situations

$$\begin{cases} \min & \text{Estimation error} \\ \text{s.t.} & \text{Model requirement constraints} \end{cases}$$

Example: Container design

A soft drink manufacturer would like to produce a cylindrical can that would hold 330 mL of liquid. The manufacturer wants to decide the dimension of this can so that the material used to make this can is minimized.

Example: Least-square solution

Suppose that we are solving a linear system $Ax = b$. If the system has a solution $\bar{x}$, then this $\bar{x}$ is also a minimizer of the function

$$f(x) = \frac{1}{2} \|Ax - b\|^2 .$$

Minimizing this function f is also beneficial when $Ax = b$ has no solution. In such situation, we are looking instead of the *least-square solution*, i.e. the point $\bar{x}$ that minimizes the difference between Ax and b.

Example: Linear regression

Suppose that at each controlled state vectors

$$x_1, \cdots, x_m \in \mathbb{R}^n,$$

we have recorded a corresponding set of observed values

$$y_1, \cdots, y_m \in \mathbb{R}$$

where it is known that $y_i \sim x_i$ linearly, i.e.

$$y_i \approx a^\top x_i + b \qquad (\forall i = 1, \cdots, m).$$

Then the linear regression aims at finding the best affine approximation $r(x) = a^\top x + b$ to all y_i's, where $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

Example: Portfolio optimization (1/2)

A man has \$20,000 worth for investment and he is interested to invest in four different financial instruments as follows.

Choice 1. Buy *stock X* which is selling at \$20 per share.

Choice 2. Purchase a *European call* options to buy a share of stock X at \$15 in exactly 6 months time. The options are selling for \$10.

Choice 3. Raise more funds for investment by selling the European call options above.

Choice 4. Purchase a 6-month riskless zero-coupon bonds having a face value of \$100 at the price of \$90.

Example: Portfolio optimization (2/2)

This man has determined that there are three *equally likely* scenarios that may occur to the stock X, namely

Scenario 1. Stock X sells for \$20 per share in 6 months.

Scenario 2. Stock X sells for \$40 per share in 6 months.

Scenario 3. Stock X sells for \$12 per share in 6 months.

Due to the risks involved, there is a margin on the total number of European call options that you can sell, in this case set to 500. Also, a person is limited to the maximum of 5000 calls.

The aim of this man is to make an investment plan for the available choices.

Tools

Tools

Vectors and Matrices

Vectors

Since optimization problems usually depends on more than one variable, we capture all the variables that we need to decide upon into a single array called a vector.

A vector x of n entries (or components) is represented by

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = (x_1, x_2, \cdots, x_n).$$

We write $x \in \mathbb{R}^n$ to inform that x is a vector of n entries.

In the case $n = 1$, a vector $x \in \mathbb{R}^1 = \mathbb{R}$ contains just a single number, which we will call a *scalar*.

Vector algebras

We may do some algebras with vectors.

Let $x, y \in \mathbb{R}^n$ with $x = (x_1, \cdots, x_n)$ and $y = (y_1, \cdots, y_n)$, then their addition and subtraction are defined componentwise:

$$x \pm y = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \pm \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 \pm y_1 \\ \vdots \\ x_n \pm y_n \end{bmatrix}.$$

We can also multiply a scalar $\alpha \in \mathbb{R}$ to a vector $x \in \mathbb{R}^n$ by broadcasting α over each entries of x:

$$\alpha x = \alpha \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \alpha x_1 \\ \vdots \\ \alpha x_n \end{bmatrix}.$$

Lengths and distances

The length of a given vector $x \in \mathbb{R}^n$ can be computed by the (Euclidean) norm $\|\cdot\|$:

$$\|x\| = \sqrt{x_1^2 + \cdots + x_n^2} = \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}.$$

The distance between two vectors $x, y \in \mathbb{R}^n$ can be calculated by the norm of their difference:

$$\|x - y\| = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} = \left(\sum_{i=1}^{n} (x_i - y_i)^2\right)^{\frac{1}{2}}.$$

Matrices

As a vector can be regarded as a 1-D array of numbers, we define a matrix to be a 2-D array of numbers.

A matrix A of dimension $m \times n$, having m rows and n columns, is usually represented by

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = [a_{ij}]_{\substack{i=1,\cdots,m \\ j=1,\cdots,n}} = [a_{ij}]_{m \times n}.$$

We write $A \in \mathbb{R}^{m \times n}$ as a shorthand for A is an $m \times n$ matrix.

We can view a vector $x \in \mathbb{R}^n$ as a matrix of dimension $n \times 1$.

Matrix algebra

The addition and subtraction of two matrices with the same dimension can be done in a componentwise fashion, and the scalar multiplication to a matrix can be done similarly by broadcasting.

To multiply two matrices A and B, we require the *width* of A to match with the *height* of B. If $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times r}$, then their product $AB = [c_{ij}]_{m\times r}$ is an $m \times r$ matrix whose entry c_{ij} is defined by

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{in} b_{nj}.$$

In general, $AB \neq BA$ even when both are defined (i.e. $m = n = r$).

Classes of matrices

A matrix $A \in \mathbb{R}^{m \times n}$ is called:

- *square* if $m = n$.
- *symmetric* if $A^\top = A$.
- *identity* if it is square and $a_{ij} = 0$ if $i \neq j$ and $a_{ij} = 1$ if $i = j$. An identity matrix is usually denoted by I.
- *invertible* if it is square and there is another matrix, denoted by A^{-1}, in which $AA^{-1} = A^{-1}A = I$. A matrix A is invertible $\iff \texttt{det}(A) \neq 0$[1].
- *positive semidefinite* ($A \succeq 0$) if it is symmetric and $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^n$.
- *positive definite* ($A \succ 0$) if it is symmetric and $x^\top Ax > 0$ for all nonzero $x \in \mathbb{R}^n$.

[1] We shall skip the definition of $\texttt{det}(A)$ as it is complicated for $n \times n$ matrices

Positivity criteria

It is not easy to check the positivity of a given matrix A. However, we have the following more practical approach.

A symmetric matrix A is

- positive semidefinite $\iff$ all eigenvalues of A are ≥ 0, and
- positive definite $\iff$ all eigenvalues of A are > 0.

Tools

Vector calculus

Multivariate functions

A function in this lecture would accepts not only a single variable x, but on several variables $x_1, x_2, \cdots, x_n$.

It is convenient to deliver all the variables above in a single package $x = (x_1, x_2, \cdots, x_n)$ as a vector in $\mathbb{R}^n$.

The functions we use throughout our discussion will generally be of the form $f(x) = f(x_1, x_2, \cdots, x_n)$.

First-order derivatives

The derivative depicts the rate of change / slope of the function at a given point x in certain directions.

The *partial derivative* of f w.r.t. the variable x_i is denoted by $\frac{\partial f}{\partial x_i}$.

The *gradient* of f is then the vector containing all of its partial derivatives, i.e.

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \cdots, \frac{\partial f}{\partial x_n}(x) \right) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}.$$

Second-order derivatives

The second-order partial derivatives of f are all the partial derivatives $\dfrac{\partial^2 f}{\partial x_j \partial x_i} = \dfrac{\partial}{\partial x_j}\dfrac{\partial f}{\partial x_i}$ for $i, j = 1, \cdots, n$.

Since the second-order partial derivatives have double indices, it is natural to pack them in a matrix, rather than a vector. Such a matrix is called the *Hession* of f:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1}(x) & \frac{\partial^2 f}{\partial x_2 x_1}(x) & \cdots & \frac{\partial^n f}{\partial x_n x_1}(x) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \frac{\partial^2 f}{\partial^2 x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_n x_2}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & \frac{\partial^2 f}{\partial x_2 x_n}(x) & \cdots & \frac{\partial^2 f}{\partial^2 x_n}(x) \end{bmatrix}.$$

If all the second-order partial derivatives of f are continuous, then $\dfrac{\partial^2 f}{\partial x_j \partial x_i} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$. This implies that $\nabla^2 f(x)$ is a symmetric matrix.

Principles of Unconstrained Optimization

Optimality conditions

All principles of optimization relies on calculus.

Necessity: $\bar{x} \in LMin(f) \implies \nabla f(\bar{x}) = 0$.

Sufficiency: $[\nabla f(\bar{x}) = 0 \wedge \nabla^2 f(\bar{x}) \succ 0] \implies \bar{x} \in LMin(f)$.

To find a point $\bar{x} \in LMin(f)$:
Filter: Solve for critical points $\nabla f(\bar{x}) = 0$.
Confirm: Check the positivity $\nabla^2 f(\bar{x}) \succ 0$ at the critical points.

Simplification: Convex case

Convex functions are those functions f whose Hessian satisfies $\nabla^2 f(x) \succeq 0$ for all $x \in \mathbb{R}^n$.

When f is convex, then

$$LMin(f) = Min(f)$$

and also

$$\bar{x} \in Min(f) \iff \nabla f(\bar{x}) = 0.$$

Quadratic functions

A quadratic function $f(x)$ of n variables takes the form

$$f(x) = x^\top Ax + b^\top x + c \qquad (\forall x \in \mathbb{R}^n)$$

where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, and $c \in \mathbb{R}$ are given coefficients.

Minimizing a quadratic function is simple as we may derive that

$$\nabla f(x) = (A + A^\top)x + b \quad \text{and} \quad \nabla^2 f(x) = A + A^\top.$$

Especially when A is symmetric, we get

$$\nabla f(x) = 2Ax + b \quad \text{and} \quad \nabla^2 f(x) = 2A.$$

Note that the Hessian of a quadratic function is always a constant.

Polynomial fitting

We demonstrate minimizing a quadratic function through a practical problem of fitting a polynomial to a collection of data.

Suppose we have m samples with $y_i \sim x_i$. We shall try to fit a polynomial of degree n to this set of samples, i.e. find the coefficients a_i in which

$$y_i \approx a_n x_i^n + a_{n-1} x_i^{n-1} + \cdots + a_1 x_i + a_0,$$

Some algorithms

General rules

The following is the general scheme for minimizing a function f.

GENERAL SEARCH SCHEME.

```
**Initialization:**
Pick a start point $x^0 \in \mathbb{R}^n$.
Set $k \leftarrow 0$.
**While:** $\nabla f(x^k) \neq 0$;
[line search] determine $d^k \in \mathbb{R}^n$ and $t_k > 0$.
[update] $x^{k+1} \leftarrow x^k + t_k d^k$.
[update] $k \leftarrow k + 1$.
```

Note 1. There is no ‘one size fits all’ sort of algorithms.
Note 2. Convergence of algorithms is usually guaranteed under strong assumptions.
Note 3. In practice, not too many people cares about the above facts!

Line search rules

After a descent direction d^k is decided, the step size t_k can then be determined by several rule:

- **Constant step-size rule.** Take some $\bar{t} > 0$ and let $t_k := \bar{t}$ for all $k \in \mathbb{N}$.
- **Vanishing step-size rule.** Choose *a priori* a decreasing positive real sequence (t_k) in which $t_k \to 0$.
- **Exact linesearch rule.** For each $k \in \mathbb{N}$, take $t_k = \operatorname{argmin}_{t>0} f(x^k + td^k)$.
- **Armijo's backtracking linesearch rule.** Choose *a priori* an acceptable rate of descent $\alpha \in (0,1)$, an initial step length $s > 0$, and a decremental ratio $\beta \in (0,1)$. For each $k \in \mathbb{N}$, define $t_k := \beta^i s$ (here β is raised to the power i), where $i \in \mathbb{N} \cup \{0\}$ is the least integer satisfying

$$f(x^k + s\beta^i d^k) \leq f(x^k) + \alpha s \beta^i \langle \nabla f(x^k), d^k \rangle.$$

Gradient descent methods

Gradient descent methods refer to the class of algorithms where

$$d^k = -\nabla f(x^k),$$

i.e. the steepest descent direction. The performance varies upon the line search rules as well as the behavior of the objective function itself.

Also note that in nonconvex cases, the algorithm is only capable of approximating a critical point.

The performance of Gradient descent methods is known to be related to the *condition number* which is the ratio of the maximum and minimum eigenvalues of the Hessians.

Gradeint descent methods

GRADIENT DESCENT METHOD.

```
**Initialization:**
Pick a start point $x^0 \in \mathbb{R}^n$.
Set $k \leftarrow 0$.
**While:** $\nabla f(x^k) \neq 0$;
[line search] determine $t_k > 0$.
[update] $x^{k+1} \leftarrow x^k - t_k \nabla f(x^k)$.
[update] $k \leftarrow k + 1$.
```

Newton’s method

The Newton’s method was orignally designed to solve a nonlinear equation

$$F(\bar{x}) = 0.$$

With $F = \nabla f$, the Newton’s method reduces to the problem of finding a critical point $\nabla f(\bar{x}) = 0$.

The Newton’s method refer to the case $d^k = -[\nabla^2 f(x^k)]^{-1} \nabla f(x^k)$ and $t_k = 1$.

Newton's methods

NEWTON'S METHOD.

```
**Initialization:**
Pick a start point $x^0 \in \mathbb{R}^n$.
Set $k \leftarrow 0$.
**While:** $\nabla f(x^k) \neq 0$;
[update] $x^{k+1} \leftarrow x^k - [\nabla^2 f(x^k)]^{-1} \nabla f(x^k)$.
[update] $k \leftarrow k + 1$.
```

In the first update step, one needs to calculate $[\nabla^2 f(x^k)]^{-1}$ which can be expensive. Hence, we may solve the following linear system instead to get x^{k+1}:

$$\nabla^2 f(x^k)(x^{k+1} - x^k) = -\nabla f(x^k).$$

The expensive computation of $\nabla^2 f(x^k)$ is further treated in the so-called *quasi-Newton methods*.

Conjugate gradient methods

Conjugate gradient methods take advantage of the geometric feature of the objective function, so that large condition number does not affect the convergence.

Another great feature about the conjugate gradient method is that it is guaranteed for to converge within n steps for quadratic functions of n variables with $A \succ 0$.

Conjugate gradient methods

For simplicity, we let

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x.$$

Then we get $\nabla f(x) = Ax - b = r(x)$ and $\nabla^2 f(x) = A$.

Conjugate gradient methods

CONJUGATE GRADIENT ALGORITHM.

```
Initialization:
Pick a start point $x^0 \in \mathbb{R}^n$.
Set $r^0 \leftarrow Ax^0 - b$, $d^0 \leftarrow -r^0$, $k \leftarrow 0$.
While: $r^k \neq 0$;
Update:
$t_k \leftarrow -\frac{(r^k)^\top d^k}{(d^k)^\top A d^k}$.
$x^{k+1} \leftarrow x^k + t_k d^k$.
$r^{k+1} \leftarrow Ax^{k+1} - b$.
$\beta_{k+1} \leftarrow \frac{(r^{k+1})^\top A d^k}{(d^k)^\top A d^k}$.
$d^{k+1} \leftarrow -r^{k+1} + \beta_{k+1} d^k$.
$k \leftarrow k + 1$.
```

Note. The Conjugate gradient method actually works with general convex functions, with some modifications to the scheme.

Some pros and cons.

Gradient descent methods.
Pros: Simple and light. Requires only the gradient information.
Cons: Can be slow to the level of being useless. Relies very largely on the condition numbers.

Newton's method.
Pros: Fast, when it works.
Cons: Expensive to compute. Convergence is difficult to analyze and usually works locally.

Conjugate gradient methods.
Pros: Light weight and relatively fast.
Cons: Very sensitive that sometimes it behaves differently from the theory. Convergence behavior is strange for non-quadratic functions.

Quasi-Newton methods

The idea of Quasi-Newton methods is to replace the exact computation of the Hessian $\nabla^2 f(x^k)$ with an approximation B_k, which will be updated at each iterate. There are various approaches, but the most famous one would be the one of Broyden-Fletcher-Goldfarb-Shanno (BFGS) and its original predecessor Davidon-Fletcher-Powell (DFP).

BFGS method

BFGS METHOD.

```
Initialization:
Pick a start point $x^0 \in \mathbb{R}^n$ and an initial Hessian approximation $H_0$.
$k \leftarrow 0$.
While: $\nabla f(x^k) \neq 0$;
    Update:
    $d^k \leftarrow -H_k \nabla f(x^k)$.
    Get $t_k > 0$ from the Wolfe's linesearch rule.[2]
    $x^{k+1} \leftarrow x^k + t_k d^k$.
    $s^k \leftarrow x^{k+1} - x^k$.    $y^k \leftarrow \nabla f(x^{k+1}) - \nabla f(x^k)$.    $\rho_k \leftarrow \frac{1}{(y^k)^\top s^k}$.
    $H_{k+1} \leftarrow (I - \rho_k s^k (y^k)^\top) H_k (I - \rho_k y^k (s^k)^\top) + \rho_k s^k (s^k)^\top$.
    $k \leftarrow k + 1$.
```

[2] In practice, people prefer to use the fixed step $t_k = 1$.

Wolfe's linesearch rule

One may approach the Wolfe's linesearch rule as in Armijo's backtracking with an additional parameter $0 < \alpha < \gamma < 1$ curvature condition to be satisfied:

$$\nabla f(x^k + s\beta^i d^k)^\top d^k \geq \gamma \nabla f(x^k)^\top d^k.$$

Again, no perfect method.

BFGS method improves the Newton's method by replacing the exact Hessian computation with a more simplified update on its approximation H_k.

However the computation of H_k is not totally cheap and in fact quite tense in storage and evaluation, in the order $O(n^2)$.

Conclusion and remarks

Conclusion and remarks

- Optimality (local) is calculated through first- and second-order derivatives.
- Global optimality is only guaranteed for convex functions.
- Algorithms are not always convergent, and not perfect for all problems.
- If convergent, they tend to go to critical points.

Thank you.