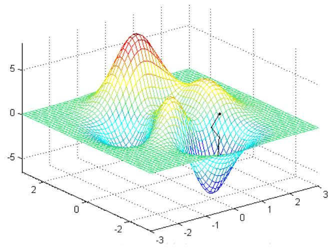
# Big Data Technology and its Applications




## Mathematics foundation

张宁 ningzhang@tsinghua.edu.cn

# Outline

- Linear programming
- Non-linear programming
- Karush–Kuhn–Tucker (KKT) conditions

# Introduction to Linear programming

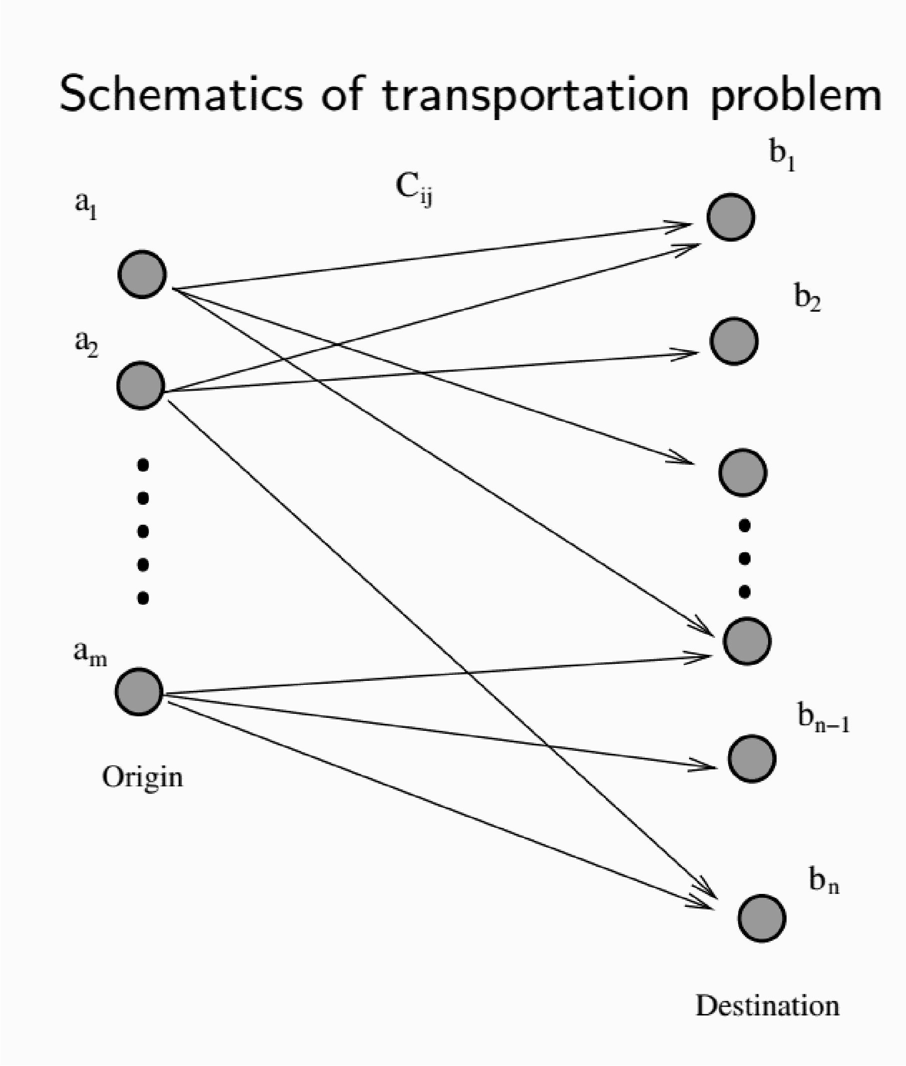
# Example: Transportation problem

- The objective consists in **minimizing transportation cost** of a given commodity from a number of **sources** or **origins** (e.g. factory, manufacturing facility) to a number of **destinations** (e.g. warehouse, store).

- Each source has a limited supply (i.e. maximum number of products that can be sent from it)

- Each destination has a demand to be satisfied (i.e. minimum number of products that need to be shipped to it).

- The cost of shipping from a source to a destination is directly proportional to the number of units shipped.



# Example: Transportation problem

- Formulation

$$\min s = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

subject to the constraints:

$$\sum_{i=1}^{m} x_{ij} \geq b_j, \qquad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} \leq a_i, \qquad i = 1, \ldots, m$$

$$x_{ij} \geq 0, \qquad \forall i, j$$

where $a_i$ is the supply of $i$-th origin, $b_j$ is the demand of the $j$-th destination, $x_{ij}$ is the amount of shipment from source $i$ to destination $j$ and $c_{ij}$ is the corresponding unit transportation cost from $i$ to $j$.

# Linear programming definition

- If the minimized (or maximized) function and the constraints are all in **linear form**, this type of optimization is called linear programming (LP).

linear form: $a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b$

- Transportation problem is a typical linear programming problem.

linear objective: $$\min s = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

linear constraints: $$\begin{cases} \sum_{i=1}^{m} x_{ij} \geq b_j, & j = 1, \ldots, n \\ \sum_{j=1}^{n} x_{ij} \leq a_i, & i = 1, \ldots, m \\ x_{ij} \geq 0, \ \forall i, j \end{cases}$$

投票 最多可选1项

设置

Does optimal solution of Linear programming has a closed form?

A Yes

B No

提交

# Standard form of LP

- The minimized function will always be

$$\min \; \boldsymbol{c}^T \boldsymbol{x} \qquad \text{(or max)}$$

- The constraints are equality constraints and variables are positive.

$$\begin{aligned} \boldsymbol{Ax} &= \boldsymbol{b} \\ \boldsymbol{x} &\geq \boldsymbol{0} \end{aligned} \quad \Longleftrightarrow \quad \begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\ \qquad x_i \geq 0 \end{cases}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^{\boldsymbol{m}}$, $\mathbf{rank}(\boldsymbol{A}) = \boldsymbol{m} \leq \boldsymbol{n}$, $\mathbf{b} \geq \mathbf{0}$

# Standard form of LP

- For general form of constraints, how to transform to standard form?

Type I: "≤" type constraint

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1$$

Introduce slack variable

$x_{n+1}$

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{n+1} = b_1$$

$$x_{n+1} \ge 0$$

Type II : "≥" type constraint

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \ge b_1$$

Introduce surplus variable

$x_{n+1}$

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - x_{n+1} = b_1$$

$$x_{n+1} \ge 0$$

- If some of $b_i < 0$ in the primitive form, we can time $-1$ to both sides at first and introduce the slack and surplus variables again.

# Fundamental theorem for LP

- For the standard form $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $n$ is called dimension, $m$ is called order, variables $\mathbf{x}$ satisfying constraints are called feasible solution.

- Suppose $\mathbf{rank}(\boldsymbol{A}) = \boldsymbol{m}$ , and the first $m$ columns of $\boldsymbol{A}$ are linearly independent, i.e.

$$\boldsymbol{B} = \left(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m\right)$$

is nonsingular, where $\boldsymbol{a}_i = (a_{1i}, a_{2i}, \ldots, a_{mi})$T . Then call ***B*** a basis.

- Then the original constraints can be rewritten as:

$$\boldsymbol{Ax} = \boldsymbol{b} \quad \Rightarrow \quad [\boldsymbol{B} \quad \boldsymbol{N}]\begin{bmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{bmatrix} = \boldsymbol{b} \quad \Rightarrow \quad \boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b} - \boldsymbol{B}^{-1}\boldsymbol{N}\boldsymbol{x}_N$$

# Fundamental theorem for LP

- As $\mathbf{rank}(\boldsymbol{A}) = \boldsymbol{m}$, we could simply let $\boldsymbol{x}_N = 0$ and get $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{B}^{-1}\boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}$

- We call $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{B}^{-1}\boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}$ a basic solution with respect to basis ***B***

- If a basic solution is also a feasible solution $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{B}^{-1}\boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} \geq 0$ , it is called a basic feasible solution.

- $x_i$ corresponding to column indices in ***B*** are called basic variable. The others are called non-basic variables.

# Fundamental theorem for LP: Example

- Linear programming example:

$$\begin{aligned} &\min -10x_1 - 11x_2 \\ &3x_1 + 4x_2 \le 9 \\ &5x_1 + 2x_2 \le 8 \\ &x_1 - 2x_2 \le 1 \\ &x_i \ge 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} &\min -10x_1 - 11x_2 \\ &3x_1 + 4x_2 + x_3 \phantom{+ x_4 + x_5} = 9 \\ &5x_1 + 2x_2 + \phantom{x_3 +} x_4 \phantom{+ x_5} = 8 \\ &x_1 - 2x_2 + \phantom{x_3 + x_4 +} x_5 = 1 \\ &x_i \ge 0 \end{aligned}$$

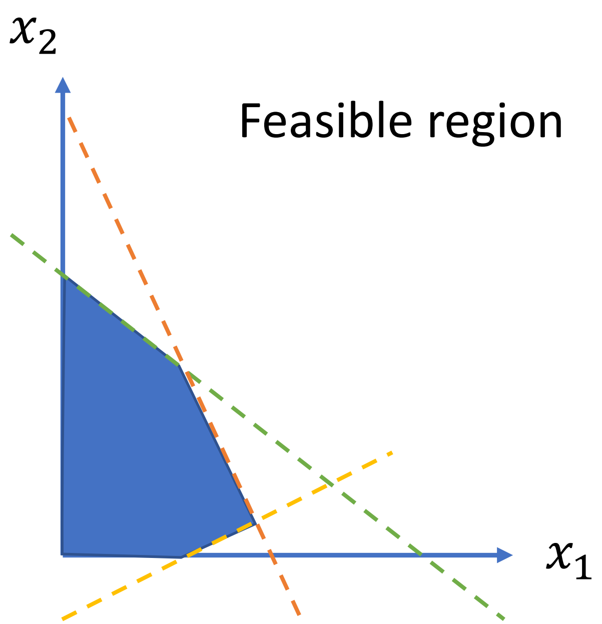
- Choose $\boldsymbol{B} = (\boldsymbol{a}_3, \boldsymbol{a}_4, \boldsymbol{a}_5) = \boldsymbol{I}_{3\times 3}$ , then B is a basis.

- $\boldsymbol{x} = (0,0,9,8,1)$ is a basic solution. It satisfies the constraint $x \ge 0$, thus is a basic feasible solution. $x_3, x_4, x_5$ are basic variables.

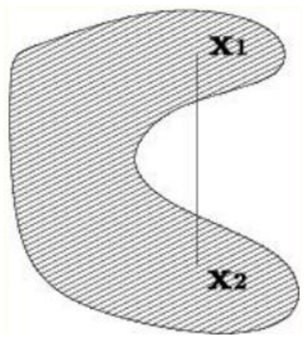
# Fundamental theorem for LP

- The set of all the feasible solutions are called feasible region.

$$\min -10x_1 - 11x_2$$
$$3x_1 + 4x_2 \leq 9$$
$$5x_1 + 2x_2 \leq 8$$
$$x_1 - 2x_2 \leq 1$$
$$x_i \geq 0$$



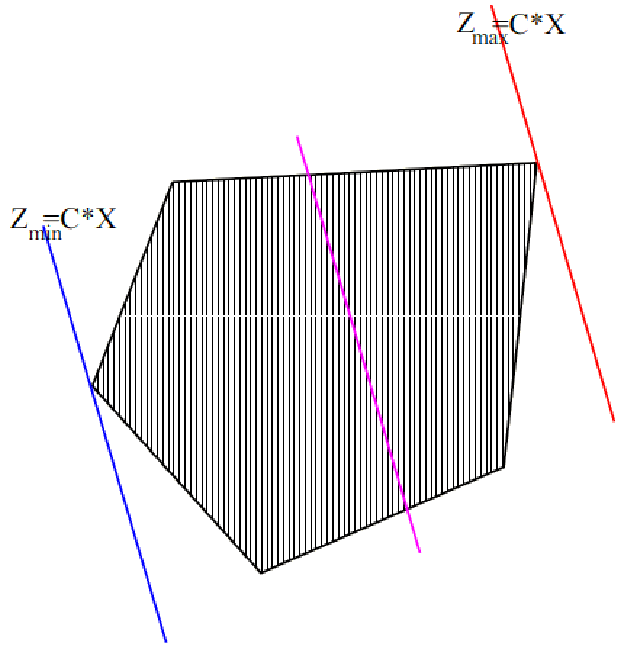

- A convex set $S$ means for any $\boldsymbol{x_1}, \boldsymbol{x_2} \in \boldsymbol{S}$ and $\lambda \in [0,1]$ , then $\boldsymbol{x} = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in S$. A non-convex set is shown here.




- The vertices of a convex set are called extreme points.

# Fundamental theorem for LP

- **Theorem 1:** Optimizing a linear objective function $c^T x$ is achieved at the extreme points in the convex feasible region if the feasible solution set is not empty and the optimum is finite.



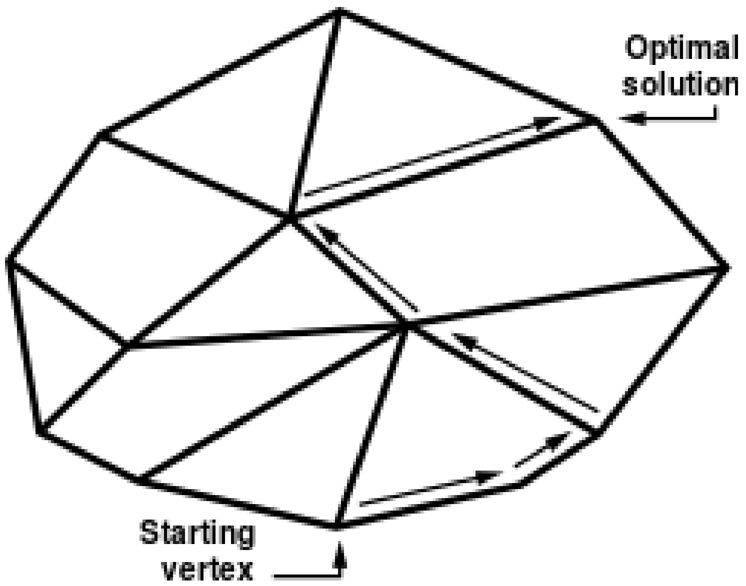

- **Theorem 2:** A point in the feasible solution set is a extreme point if and only if it is a basic feasible solution.

# Simplex method for LP

- Simplex method is first proposed by G.B. Dantzig in 1947.

- Simply searching for all of the basic solution is not applicable because the whole number is $C_n^m$

- Basic idea of simplex: Give a rule to transfer from one extreme point to another such that the objective function is decreased. This rule must be easily implemented.



# Simplex method for LP

- First suppose the standard form is $\boldsymbol{Ax}=\boldsymbol{b},\ \ \boldsymbol{x}\geq\boldsymbol{0}$

- One canonical form (标准型) is to transfer a coefficient submatrix into $\boldsymbol{I}_m$ with Gaussian elimination. For example

$$\boldsymbol{x}=\left(x_1,x_2,x_3,x_4,x_5\right)$$

$$\left(\boldsymbol{A}\,|\,\boldsymbol{b}\right)=\left(\begin{array}{cc|ccc|c} 3 & 4 & 1 & & & 9 \\ 5 & 2 & & 1 & & 8 \\ 1 & -2 & & & 1 & 1 \end{array}\right)$$

- then it is a canonical form for $x_3\ x_4, x_5$. They are basic variables and the extreme point is

$$\boldsymbol{x}=\left(0,0,9,8,1\right)$$

the extreme point $\boldsymbol{x}=\begin{bmatrix}\boldsymbol{B}^{-1}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix}$ $\longrightarrow$ $\begin{pmatrix}x_3\\x_4\\x_5\end{pmatrix}=\begin{pmatrix}1&&\\&1&\\&&1\end{pmatrix}^{-1}\begin{pmatrix}9\\8\\1\end{pmatrix}$

# Simplex method for LP

- Now suppose $\boldsymbol{A}$ is in canonical form as the last example, then we transfer from one basic solution to another.

- Choose $x_2$ to enter the basis and $x_3$ to leave the basis

$$
(\boldsymbol{A}\,|\,\boldsymbol{b}) = \left(\begin{array}{ccccc|c} 3 & 4 & 1 & & & 9 \\ 5 & 2 & & 1 & & 8 \\ 1 & -2 & & & 1 & 1 \end{array}\right) \to \left(\begin{array}{ccccc|c} 3/4 & 1 & 1/4 & & & 9/4 \\ 5 & 2 & & 1 & & 8 \\ 1 & -2 & & & 1 & 1 \end{array}\right) \to \left(\begin{array}{ccccc|c} 3/4 & 1 & 1/4 & & & 9/4 \\ 14/4 & & -2/4 & 1 & & 14/4 \\ 10/4 & & 2/4 & & 1 & 11/4 \end{array}\right)
$$

- then it is a canonical form for $x_2$, $x_4$ and $x_5$. The basic solution is $x = (0, \frac{9}{4}, 0, \frac{14}{4}, \frac{11}{4})$. It is also a extreme point.

- The transferred basic solution may be not feasible in general.
  1. How to make the transferred basic solution feasible?
  2. How to make the objective function decreasing after transfer?

# How to make the transferred basic solution feasible?

- Assumption: All of the basic feasible solutions are non-degenerate.
  i.e. if $\boldsymbol{x} = (x_1, x_2, \ldots, x_m, 0, \ldots, 0)$ is a basic feasible solution, then $x_i > 0$.

- Suppose the basis is $\{\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_m}\}$ initially, and select $\boldsymbol{a}_k (k > m)$ enter the basis.
  Suppose:

$$\boldsymbol{a}_k = \sum_{i=1}^{m} y_{ik} \boldsymbol{a}_i$$

- then for any $\epsilon > 0$

$$\varepsilon \boldsymbol{a}_k = \sum_{i=1}^{m} \varepsilon y_{ik} \boldsymbol{a}_i$$

- Suppose $\boldsymbol{x}$ is a basic feasible solution initially.

$$\sum_{i=1}^{m} x_i \boldsymbol{a}_i = \boldsymbol{b}$$

# How to make the transferred basic solution feasible?

- Then we have

$$\sum_{i=1}^{m}(x_i - \varepsilon y_{ik})\boldsymbol{a}_i + \varepsilon \boldsymbol{a}_k = \boldsymbol{b}$$

- Because $x_i > 0$, if $\epsilon > 0$ is small enough,

$$x = \left(x_1 - \varepsilon y_{1k}, x_2 - \varepsilon y_{2k}, \ldots, x_m - \varepsilon y_{mk}, 0, \ldots, 0, \underset{\substack{\uparrow \\ \text{position } k}}{\varepsilon}, 0, \ldots 0\right)$$

is a feasible solution.

- To make it a basic solution we choose

$$\varepsilon = \min_{1 \le i \le m}\left\{\frac{x_i}{y_{ik}} \middle| y_{ik} > 0\right\} = \frac{x_r}{y_{rk}}$$

then $\tilde{x}$ is a basic feasible solution, and we can let the selected $\boldsymbol{a}_r$ leave the basis while the $\boldsymbol{a}_k$ $(k > m)$ enter the basis.

# How to make the objective function decrease after transfer?

- The aim is to choose $k$ such that the objective function decreasing after $\boldsymbol{a_k}$ enter the basis.

- Suppose the basic feasible solution is

$$\boldsymbol{x} = \left(x_{10}, x_{20}, \ldots, x_{m0}, 0, \ldots, 0\right)$$

- The value of objective function is

$$z_0 = \boldsymbol{c}_B^T \boldsymbol{x}_B = \sum_{j=1}^{m} c_j x_{j0}$$

# How to make the objective function decrease after transfer?

- For any feasible solution $\boldsymbol{x} = (x_1, x_2, \dots, x_m, x_{m+1}, \dots, x_n)$, we have:

Why? ➡ $z$

$$= \sum_{j=1}^{m} c_j (x_{j0} - \sum_{k=m+1}^{n} y_{jk} x_k) + \sum_{k=m+1}^{n} c_k x_k$$

$$x = (x_{10} - y_{1k}x_k, x_{20} - y_{2k}x_k, \dots, x_{m0} - y_{mk}x_k, 0, \dots, 0, x_k, 0, \dots 0)$$

$$\boldsymbol{a}_k = \sum_{i=1}^{m} y_{ik} \boldsymbol{a}_i$$

$$= \sum_{j=1}^{m} c_j x_{j0} + \sum_{k=m+1}^{n} c_k x_k - \sum_{k=m+1}^{n} (\sum_{j=1}^{m} c_j y_{jk}) x_k$$

$$= z_0 + \sum_{k=m+1}^{n} (c_k - \sum_{j=1}^{m} c_j y_{jk}) x_k$$

$$= z_0 + \boxed{\sum_{k=m+1}^{n} (c_k - z_k) x_k}$$

where $z_k = \boldsymbol{c}_B^T \boldsymbol{y}_k = \sum_{j=1}^{m} c_j y_{jk}$

- if there exists $k$ $(m + 1 \le k \le n)$ such that $r_k = c_k - z_k < 0$, then when $x_k$ changes from 0 to positive, the objective function will be decreased.

- Optimality Criterion: If $\forall k \; r_k \ge 0$, then it is an optimal feasible solution

# Simplex method for LP: Example

- Example

$$\min z = -(3x_1 + x_2 + 3x_3)$$
$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \le \begin{pmatrix} 2 \\ 5 \\ 6 \end{pmatrix}, \quad x \ge 0$$

- Step 1: change into standard form

$$\min z = -(3x_1 + x_2 + 3x_3)$$
$$\begin{pmatrix} 2 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 3 & 0 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 2 \\ 5 \\ 6 \end{pmatrix}, \; x_i \ge 0, i = 1,2, \ldots 6$$

# Simplex method for LP: Example

- Step 2: Choose $x_4, x_5, x_6$ as basic variables, and compute the test number

$$r_1 = c_1 - z_1 = -3, \quad r_2 = c_2 - z_2 = -1, \quad r_3 = c_3 - z_3 = -3$$

set up simplex tableau

| **Basis** | $\boldsymbol{a_1}$ | $\boldsymbol{a_2}$ | $\boldsymbol{a_3}$ | $\boldsymbol{a_4}$ | $\boldsymbol{a_5}$ | $\boldsymbol{a_6}$ | **b** |
|---|---|---|---|---|---|---|---|
| $a_4$ | 2 | 1 | 1 | 1 | 0 | 0 | 2 |
| $a_5$ | 1 | 2 | 3 | 0 | 1 | 0 | 5 |
| $a_6$ | 2 | 2 | 1 | 0 | 0 | 1 | 6 |
| $r_k$ | -3 | -1 | -3 | 0 | 0 | 0 | $z_0 = 0$ |

# Simplex method for LP: Example

- Step 3： Choose vector to enter the basis. Because $\mathrm{r}_k < 0$, k=1,2,3, any one among $a_1, a_2, a_3$ could enter the basis. We choose $a_2$ (in general, $a_1$ $or$ $a_3$ will be chosen because -3 is smaller)
- Step 4： Choose vector to leave the basis. Compute $\frac{x_{i0}}{y_{ir}}, y_{ir} > 0$

  r=2, i=4, 5, 6, we have

$$\frac{x_{40}}{y_{42}} = 2, \qquad \frac{x_{50}}{y_{52}} = 2.5, \qquad \frac{x_{60}}{y_{62}} = 3$$

  Thus $a_4$ leave the basis.
- Step 5： Perform Gaussian elimination to obtain a new canonical form for basis $a_2, a_5, a_6$and set up simplex tableau.

| Basis | $\boldsymbol{a_1}$ | $\boldsymbol{a_2}$ | $\boldsymbol{a_3}$ | $\boldsymbol{a_4}$ | $\boldsymbol{a_5}$ | $\boldsymbol{a_6}$ | **b** |
|---|---|---|---|---|---|---|---|
| $a_2$ | 2 | 1 | 1 | 1 | 0 | 0 | 2 |
| $a_5$ | -3 | 0 | 1 | -2 | 1 | 0 | 1 |
| $a_6$ | -2 | 0 | -1 | -2 | 0 | 1 | 2 |
| $r_k$ | -1 | 0 | -2 | 1 | 0 | 0 | $z_0 = -2$ |

# Simplex method for LP: Example

- Step 6： Choose vector to enter the basis. Because $r_k < 0$, k=1, 3, any one among $a_1, a_3$ could enter the basis. We choose $a_3$.
- Step 7： Choose vector to leave the basis. Compute $\frac{x_{i0}}{y_{ir}}, y_{ir} > 0$

  r=3, i=2, 5, 6, we have ($y_{i3}$>0, i=2, 5)

$$\frac{x_{20}}{y_{23}} = 2, \qquad \frac{x_{50}}{y_{53}} = 1$$

  Thus $a_5$ leave the basis.
- Step 8： Perform Gaussian elimination to obtain a new canonical form for basis $a_2, a_3, a_6$ and set up simplex tableau.

| **Basis** | $\boldsymbol{a_1}$ | $\boldsymbol{a_2}$ | $\boldsymbol{a_3}$ | $\boldsymbol{a_4}$ | $\boldsymbol{a_5}$ | $\boldsymbol{a_6}$ | **b** |
|---|---|---|---|---|---|---|---|
| $a_2$ | 5 | 1 | 0 | 3 | -1 | 0 | 1 |
| $a_3$ | -3 | 0 | 1 | -2 | 1 | 0 | 1 |
| $a_6$ | -5 | 0 | 0 | -3 | 2 | 1 | 4 |
| $r_k$ | -7 | 0 | 0 | -3 | 2 | 0 | $z_0 = -4$ |

# Simplex method for LP: Example

- Step 9： Choose vector to enter the basis. Because $r_k < 0$, k=1, 4, any one among $a_1, a_4$ could enter the basis. We choose $a_1$.
- Step 10： Choose vector to leave the basis. Compute $\frac{y_{i0}}{y_{ik}}, y_{ik} > 0$

  r=1, i=2, 3, 6, we have($y_{i1}$>0, i=2)

$$\frac{y_{20}}{y_{21}} = \frac{1}{5}$$

  Thus $a_2$ leave the basis.
- Step 11： Perform Gaussian elimination to obtain a new canonical form for basis $a_1, a_3, a_6$ and set up simplex tableau.

| Basis | $\boldsymbol{a_1}$ | $\boldsymbol{a_2}$ | $\boldsymbol{a_3}$ | $\boldsymbol{a_4}$ | $\boldsymbol{a_5}$ | $\boldsymbol{a_6}$ | b |
|---|---|---|---|---|---|---|---|
| $a_2$ | 1 | $\frac{1}{5}$ | 0 | $\frac{3}{5}$ | $-\frac{1}{5}$ | 0 | $\frac{1}{5}$ |
| $a_3$ | 0 | $\frac{3}{5}$ | 1 | $-\frac{1}{5}$ | $\frac{2}{5}$ | 0 | $\frac{8}{5}$ |
| $a_6$ | 0 | 1 | 0 | -1 | 0 | 1 | 4 |
| $r_j$ | 0 | $\frac{7}{5}$ | 0 | $\frac{6}{5}$ | $\frac{3}{5}$ | 0 | $z_0 = -\frac{27}{5}$ |

# Simplex method for LP: Example

- Step 12： Choose vector to enter the basis. Because $r_k > 0$, k=1, 3, 6, so we obtain the optimal solution z*=$-\frac{27}{5}$, and the corresponding extreme point is

$$x = (\frac{1}{5}, 0, \frac{8}{5}, 0, 0, 4)$$

**Minimal objective achieved!**

# Introduction to
# Non-linear programming

# Example: Nonlinear least squares

- Suppose we have a series of experimental data $(x_i, y_i), i = 1, \dots, m$. We wish to find parameter $\theta \in \mathbb{R}^n$ such that the remainder is minimized.

$$r_i(\theta) = y_i - f_\theta(x_i)$$

- Mathematically, the objective is to find the optimal parameter $\theta$ that minimize the error function

$$\min \quad \phi(\theta) = \frac{1}{2}\boldsymbol{r}^T(\theta)\boldsymbol{r}(\theta)$$

where $\boldsymbol{r}(\theta) = (r_1, r_2, \dots, r_m)$

- This is a nonlinear programming, as the objective contains quadratic terms.
- If the function $f_\theta$ is linear, it is called least square problem and is a convex problem

$$f_\theta(x) = \theta x$$

- If the function $f_\theta$ is nonlinear, it is called nonlinear least square problem and is usually a non-convex problem

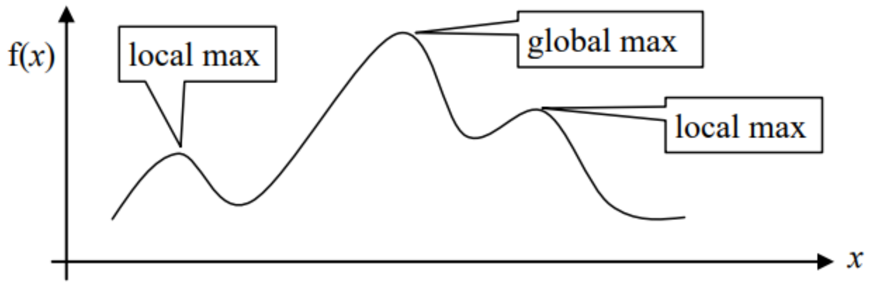
$$f_\theta(x) = \sin(\theta x)$$

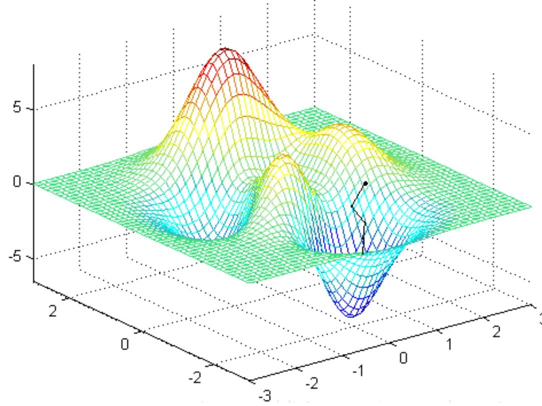
# Convex and Non-convex programming

- General form of nonlinear optimization

$$\min f(\boldsymbol{x})$$
$$g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m$$
$$h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, p$$
$$\boldsymbol{x} \in X \subset \mathbb{R}^n, \boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

- If $f(x), g(x)$ are convex function and $h(x)$ is linear function, the problem is a convex programming. Otherwise, it is a non-convex programming.
- Convex programming has only one optimum point, which is global optimum.
- Non-convex programming usually has many local optimum points.



# How to solve nonlinear programming?

- General idea 1——Iterative methods
- Object: construct sequence $\{\boldsymbol{x}_k\}_{k=1\to\infty}$ such that $\boldsymbol{x}_k$ converges to a fixed vector
- **Non-gradient method**: Golden section method, bisection method…
- **Gradient method**: make the optimum of the optimization the root of gradient equations and constraints:

$$g(x) = 0$$

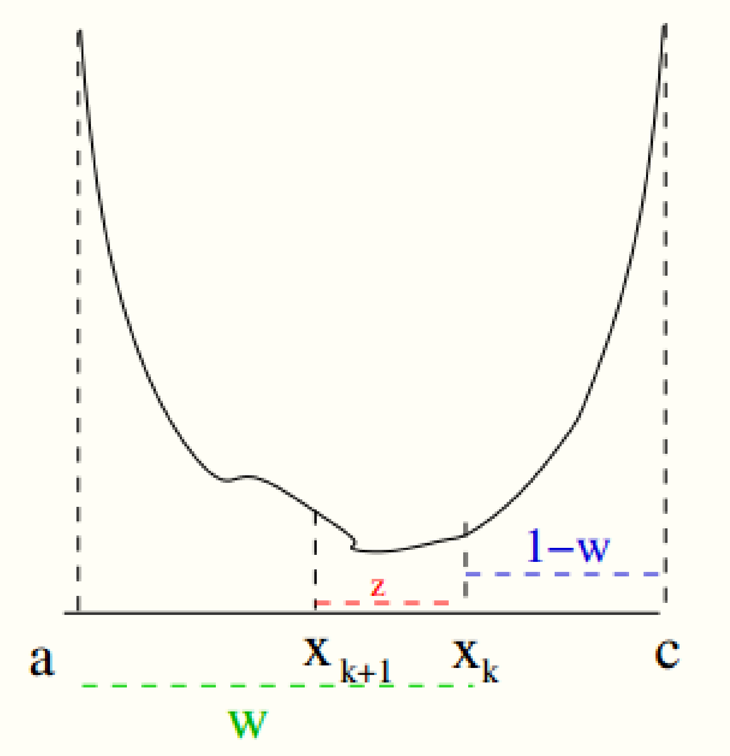
- Make another equation $x = f(x)$ that has the same solution as it, then construct

$$x_{k+1} = f\left(x_k\right)$$

- If $x_k \to x^\star$, then $x^\star = f(x^\star)$, thus the root of $g(x) = 0$ is obtained.

# Golden section method

- Suppose there is a triplet $(a, x_k, c)$ and $f(x_k) < f(a), f(x_k) < f(c)$, we want to find $x_{k+1}$ in $(a, c)$ to perform a section. Suppose $x_{k+1}$is in $(a, x_k)$.




- If $f(x_{k+1}) > f(x_k)$, then the new search interval is $(x_{k+1}, c)$;
- If $f(x_{k+1}) < f(x_k)$, then the new search interval is $(a, x_k)$.

# Golden section method

- Define

$$w = \frac{x_k - a}{c - a} \qquad 1\text{-}w = \frac{c - x_k}{c - a} \qquad z = \frac{x_k - x_{k+1}}{c - a}$$

- If we want to minimize the worst case possibility (for two cases), we must make

$$w = z + (1 - w). \quad (w > 1/2)$$

- Pay attention that *w* is also obtained from the previous stage of applying same strategy. This scale similarity implies

$$\frac{z}{w} = 1 - w$$

- We have

$$w = \frac{\sqrt{5} - 1}{2} \approx 0.618$$

- This is called Golden section method.

# Golden section method

- Golden section method is a method to find the local minimum of a function $f$.
  (Global minimum for convex function)

- Golden section method is a linear (first-order) convergence method. The contraction coefficient is $C = 0.618$.

# Steepest decent method (最速下降法)<br>Gradient Descent method (梯度下降法)

- Basic idea: Find a series of decent directions $\boldsymbol{p}_k$ and corresponding stepsize $\alpha_k$ such that the iterations

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$$
$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k).$$

- The negative gradient direction $-\nabla f$ is the “steepest” decent direction, so choose

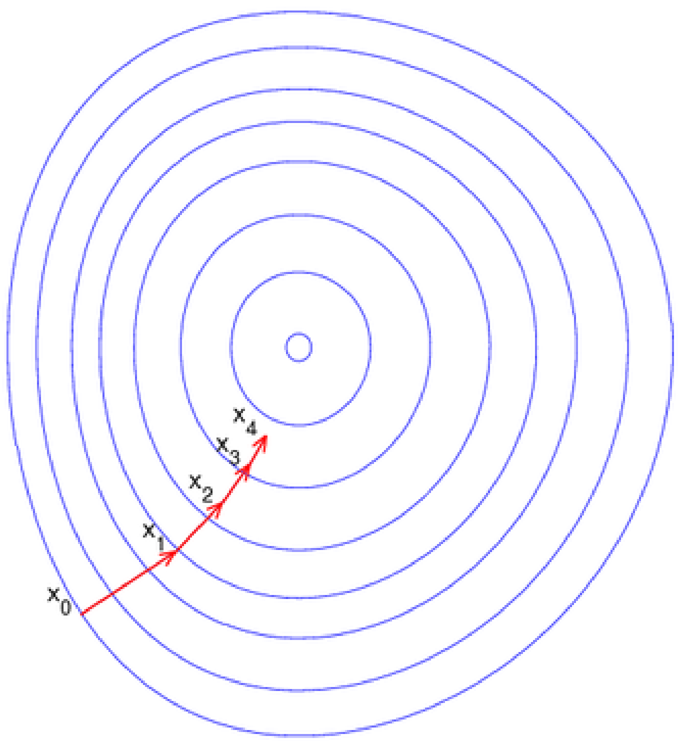
$$\boldsymbol{p}_k = -\nabla f(\boldsymbol{x}_k)$$

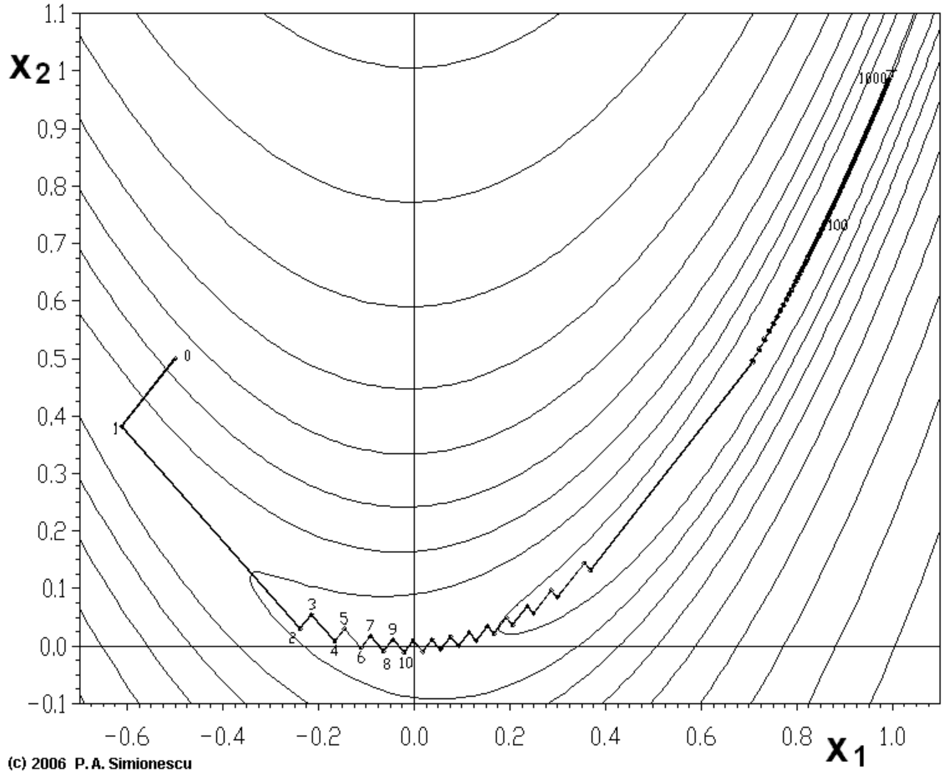
and choose $\alpha_k$ such that

$$\min_{\alpha} f\left(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k\right)$$

# Steepest decent method (最速下降法)
# Gradient Descent method (梯度下降法)

Also a linear convergence method. However, first order convergence is a bit slow.
Is there any method that converges faster?



# One dimensional Newton's method (牛顿法)

- Suppose we want to minimize $f(x)$ without any constraints

$$\min f(x)$$

- Taylor expansion at current iteration point $x_0$

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2 + \ldots.$$

- Local quadratic approximation

$$f(x) \approx g(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2$$

- Minimize $g(x)$ at $g'(x) = 0$ , then

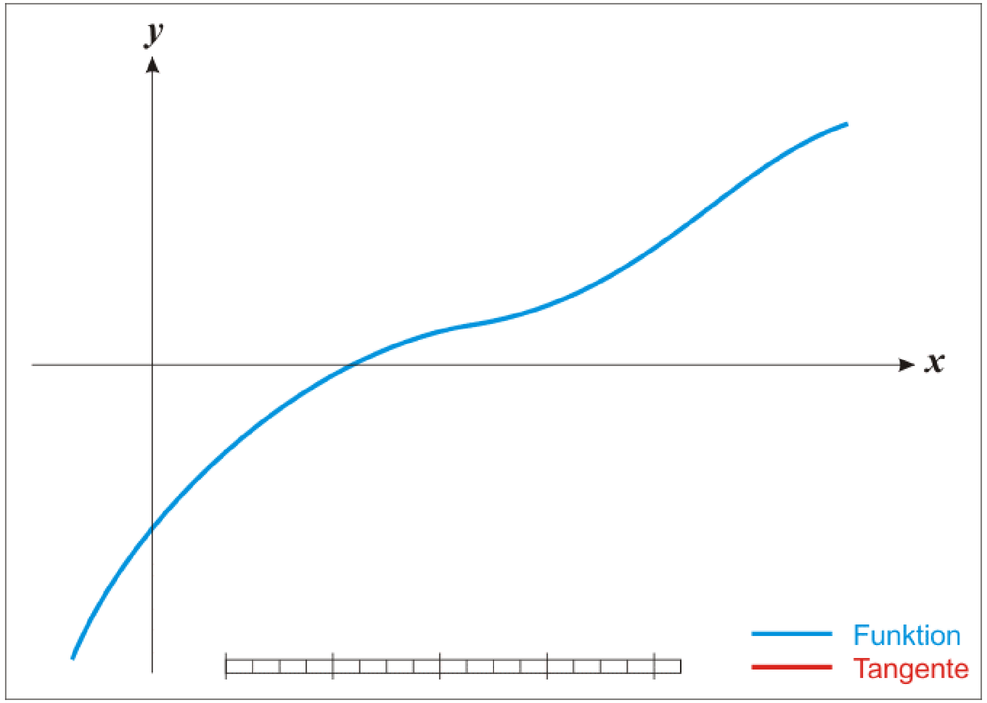
$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

- Newton's method

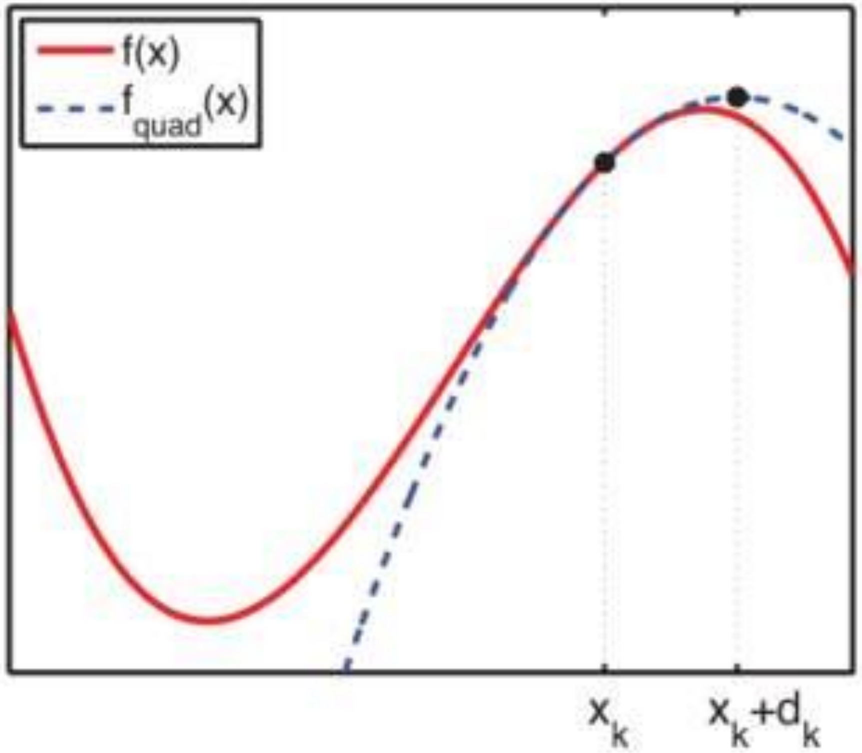
$$x_{k+1} = x_k - \frac{f'(x_0)}{f''(x_0)}$$

Newton's method converges with second order

# One dimensional Newton's method




$$g'(x) = 0$$




$$\min f(x)$$

# High dimensional Newton's method

- Suppose we want to minimize $\phi(\boldsymbol{x}), x \in \mathbb{R}^n$

$$\min \phi(\boldsymbol{x})$$

- Taylor expansion at current iteration point $\boldsymbol{x}_0$

$$\phi(\boldsymbol{x}) = \phi(\boldsymbol{x}_0) + \nabla\phi(\boldsymbol{x}_0)(\boldsymbol{x}-\boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_0)^T \nabla^2\phi(\boldsymbol{x}_0)(\boldsymbol{x}-\boldsymbol{x}_0) + \ldots$$

- Local quadratic approximation

$$\phi(\boldsymbol{x}) \approx g(\boldsymbol{x}) = \phi(\boldsymbol{x}_0) + \nabla\phi(\boldsymbol{x}_0)(\boldsymbol{x}-\boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_0)^T \boldsymbol{H}(\boldsymbol{x}_0)(\boldsymbol{x}-\boldsymbol{x}_0)$$

Hessian matrix

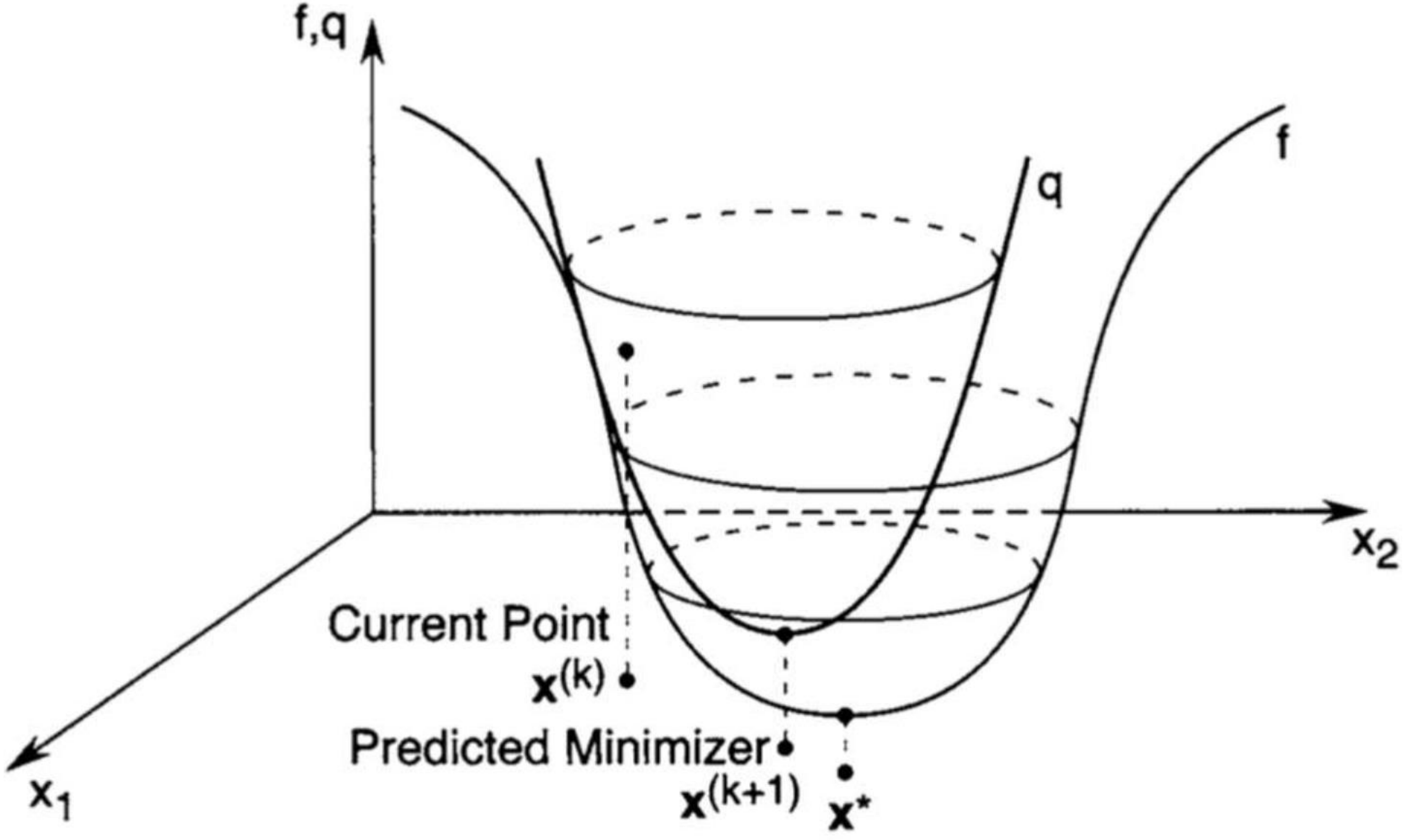
- Minimize $g(x)$ at $\nabla g(x) = 0$ , then

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \boldsymbol{H}^{-1}(\boldsymbol{x}_0)\nabla\phi(\boldsymbol{x}_0)$$

- Newton's method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{H}^{-1}(\boldsymbol{x}_k)\nabla\phi(\boldsymbol{x}_0)$$

# High dimensional Newton's method



# How to solve nonlinear programming?

- General idea 2——find the points with gradient equals zero

$$\frac{\partial f}{\partial x} = 0$$

- Difficulty 1: The equation $\frac{\partial f}{\partial x} = 0$ may be hard to solve. Iterative methods work!

- Difficulty 2: How to solve the constrained nonlinear optimization?

$$\min f(\boldsymbol{x})$$

$$g_i(\boldsymbol{x}) \le 0, \quad i = 1, 2, \ldots, m$$

$$h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, p$$

$$\boldsymbol{x} \in X \subset \mathbb{R}^n, \boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

# Introduction to KKT Conditions

# Karush-Kuhn-Tucker conditions

- Given a minimization problem (regardless of whether is convex or not)

$$\min f(\boldsymbol{x})$$

$$g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m$$

$$h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, p$$

$$\boldsymbol{x} \in X \subset \mathbb{R}^n, \boldsymbol{x} = (x_1, x_2, \ldots, x_n)$$

- We define the Lagrangian function:

$$L(x,u,v) = f(x) + \sum_{i=1}^{m} \mu_i g_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

- The Karush-Kuhn-Tucker conditions or KKT conditions are:

$$\frac{\partial L(x,u,v)}{\partial x} = 0$$ stationarity $$\nabla f(x) + \sum_{i=1}^{m} \mu_i \nabla g_i(x) + \sum_{i=1}^{p} \nu_i \nabla h_i(x) = 0$$

$$\mu \bullet g(x) = 0$$ complementary slackness

$$g(x) \leq 0, h(x) = 0$$ primal feasibility

$$\mu \geq 0$$ dual feasibility

# KKT conditions

- For a non-convex programming:

$$x^\star \text{ is local optimum} \Rightarrow x^\star, \mu^\star, \nu^\star \text{ satisfy the KKT conditions}$$

- For a convex programming:

$$x^\star \text{ is global optimum} \Leftrightarrow x^\star, \mu^\star, \nu^\star \text{ satisfy the KKT conditions}$$

# KKT conditions: Example

$$\min J = x_1^2 + x_2^2 + x_3^2$$
$$s.t. \quad x_1 + x_2 + x_3 = 1$$
$$x_1 \le \frac{1}{2}$$

Lagrangian function $L = x_1^2 + x_2^2 + x_3^2 + \mu\left(x_1 - \frac{1}{2}\right) + \nu\left(x_1 + x_2 + x_3 - 1\right)$

$$\frac{\partial L(x,u,v)}{\partial x} = \begin{bmatrix} 2x_1 + \nu + \mu \\ 2x_2 + \nu \\ 2x_3 + \nu \end{bmatrix} = 0$$
stationarity

$$\mu\left(x_1 - \frac{1}{2}\right) = 0$$
complementary slackness

$$x_1 + x_2 + x_3 = 1$$
$$x_1 \le \frac{1}{2}$$
primal feasibility

$$\mu \ge 0$$
dual feasibility

$$x_1 = x_2 = x_3 = \frac{1}{3}$$

# Homework

In [1]:

```
from scipy.optimize import linprog
```

In [2]:

```
c = [-3, -1, -3]
A = [[2, 1, 1], [1, 2, 3], [2, 2, 1]]
b = [2, 5, 6]
```

In [3]:

```
res = linprog(c, A_ub = A, b_ub = b, options={"disp": True})
```

- Given the python code of solving LP problem, make the problem into a non-linear programing problem and solve it using python.

# Q&A