

计算复杂性

- 算法性能:
- (1) 在最坏情况下算法所表现出来的性能;
-----最坏性能
- (2) 在各种情况可能出现时, 算法所表现出来的期望性能。-----平均性能

旅行商(TSP)问题

- 一个商人欲到 n 个城市推销商品，每两个城市 i 和 j 之间的距离为 d_{ij} ，如何选择一条道路，使得商人每个城市走过一遍后回到起点且所走路径最短。
- 解：设 $x_{ij}=1$ 若商人行走的路线中包含从城市 i 到 j 的路径，否则 $x_{ij}=0$ 。

$$\left\{ \begin{array}{l} \min \sum_{i \neq j} d_{ij} x_{ij} \\ s.t. \quad \sum_{j=1}^n x_{ij} = 1, i = 1, 2, \dots, n \\ \sum_{i=1}^n x_{ij} = 1, j = 1, 2, \dots, n \\ \sum_{i,j \in S} x_{ij} \leq |S| - 1, 2 \leq |S| \leq n-1, S \subset \{1, \dots, n\} \\ x_{ij} = 1 \text{ or } 0 \end{array} \right.$$

- 可行解：用 n 个城市的一个排列表示商人按这个排列次序推销并返回起点。
- 使用枚举法求解，需要 $(n-1)!$ 次枚举。
- 以计算机1秒可以完成24个城市所有路径枚举为单位。
- 城市数 24 25 26 27 30
- 计算时间 1秒 24秒 10分 4.3小时 10.8年

可怕的指数

$$n^5, 2^n, 10^n, n!$$

- 印度国王赏赐国际象棋的发明人和进贡者
 - 64个小格子
 - 第1个放1粒米
 - 第2个放2粒米
 - 下一个格子是前面一个的2倍
 - 放满整个的棋盘
- 2000年的全部!! 地球表面(含海面)被铺满厚度是1英寸!!



$2^{64} - 1$ 指数爆炸!!

- **1. 问题与实例**

- **问题(problem)**: 需要回答的一种提问, 通常包含一些参数和取值未定的自由变量, 可以从两个方面加以描述:

- (1) 对所有参数的一般描述;
- (2) 对回答 (也称为**解**) 所需要满足的特性的描述。

- **实例(instance)**: 当对一个问题中的参数赋予特定的数值时, 如何寻找相应的回答 (解), 这种提问称为该问题的一个实例。

- 问题是对许多具体事例构成集合的一种抽象表述, 而实例就是相应问题的一种具体表现形式。

- 例：线性规划问题与实例
- 一个线性规划问题的实例是指矩阵和向量组 (A, b, c) 的某一特定取值，这些参数按照如下的结构关联在一起，描述了问题（解）所需要满足的特性。

$$\begin{array}{ll}\min & cx \\ s.t. & Ax = b \\ & x \geq 0\end{array}$$

线性规划问题是对具有上述结构的所有实例的一种抽象描述。

算法

- **算法**：是一组含义明确的简单指令。
- **一个问题**是**算法可解的(solvable)**：存在一个求解该问题的算法，只要让算法运行足够长的时间，并且保证满足算法在运行过程中所需要的存储空间，它就能求解该问题的任何一个实例。
- **停机问题**：不可能构造出一个程序来确定任意给出的程序是否会陷入无限循环。

算法复杂性

- **算法复杂性(algorithm complexity):**
描述算法的存储要求和运行时间要求，分为算法的空间复杂性和算法的时间复杂性。
- ----利用算法需要的初等运算次数来表示算法的时间复杂性。

多项式时间算法与指数时间算法

- **输入规模(input size)**: 表示一个实例所需要的字符串长度。
- 一般的, 使用 $\lceil 1 + \log_2 r \rceil$ 位二进制就可以表示任意整数 r 。

- 线性规划的输入规模为:

$$L = \lceil 1 + \log_2 m \rceil + \lceil 1 + \log_2 n \rceil + \sum_{j=1}^n \left\{ \lceil 1 + \log_2 |c_j| \rceil \right\}$$

$$+ \sum_{i=1}^m \sum_{j=1}^n \left\{ \lceil 1 + \log_2 |a_{ij}| \rceil \right\} + \sum_{i=1}^m \left\{ \lceil 1 + \log_2 |b_i| \rceil \right\}$$

$$\leq mn + m + n + 2 + \log_2 |P| \quad (P \text{ 为 } A, b, c \text{ 中所有非零数的乘积})$$

- 对应TSP，枚举算法的基本计算总次数为 $[(n-1)!]n=n!$
- 实例的二进制输入长度总量不超过
- $L=n(n-1)+\log_2|P|$
- 其中P为所有非零数 d_{ij} 的乘积。
- 假设 $S=\{d_{ij} \mid 1 \leq i, j \leq n, i \neq j\}$ 中每个数据都有上界K，则有

$$L \leq n(n-1)(1+\log_2 K)$$

- 一个求解实例 I 的算法的基本计算总次数 $C(I)$ 同实例 I 的计算机二进制输入长度 $d(I)$ 的关系常用符号 $C(I)=f(d(I))=O(g(d(I)))$ 表示，它的含义：求解实例 I 的算法的基本计算总次数 $C(I)$ 是实例输入长度 $d(I)$ 的一个函数，该函数被另一个函数 $g(x)$ 控制，即存在一个函数 $g(x)$ 和一个常数 a ，使得

$$C(I) \leq ag(d(I))$$

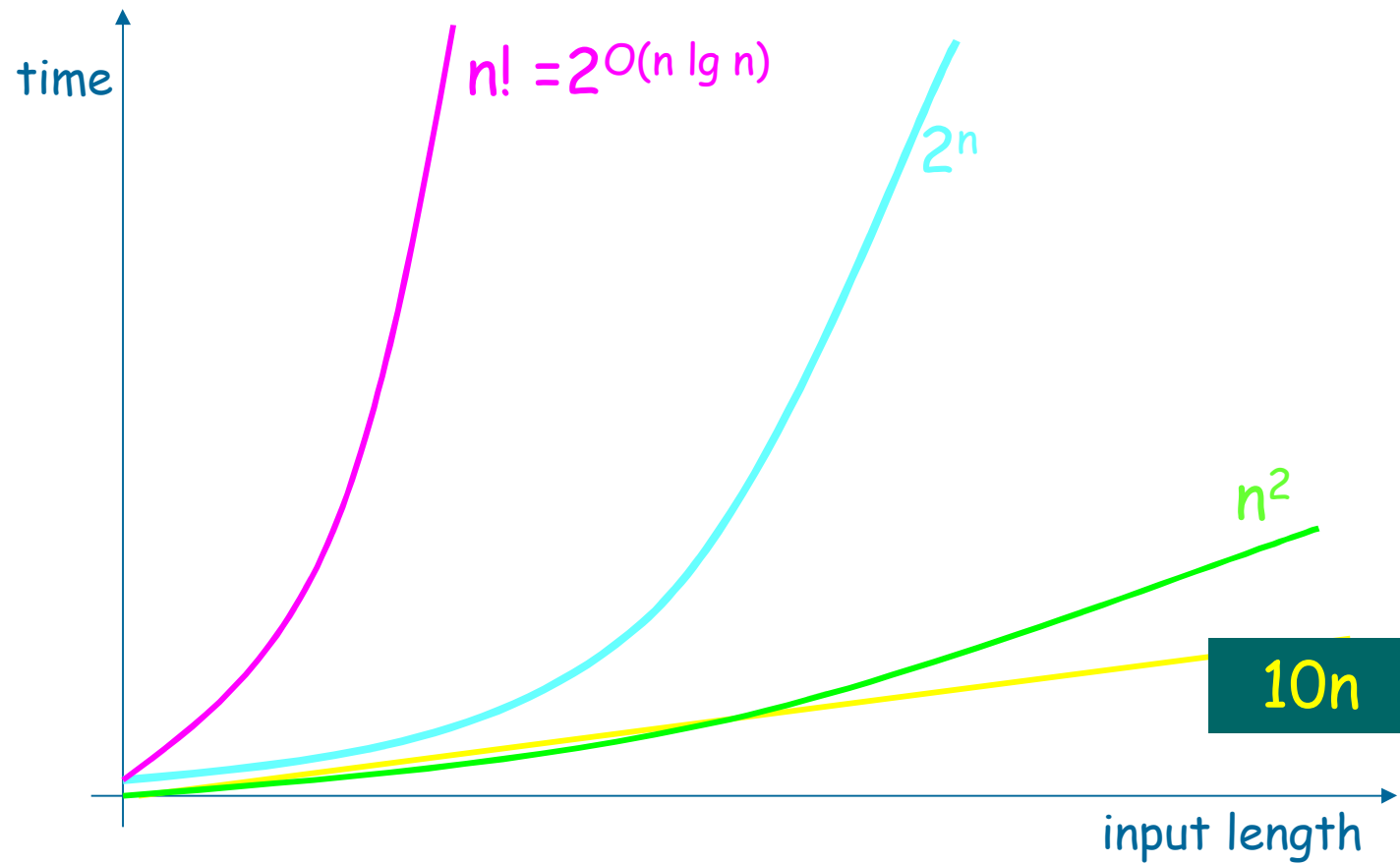
多项式时间算法与指数时间算法

- 定义：假设问题和解决该问题的一个算法已经给定，若给定该问题的一个实例 I ，存在多项式函数 $g(x)$ ，使得

$$C(I) \leq ag(d(I))$$

- 成立，则称该算法对实例 I 是多项式时间算法；若存在 $g(x)$ 为多项式函数且对该问题任意一个实例 I ，都有上式成立，则称该算法为解决该问题的多项式时间算法。
- 当 $g(x)$ 为指数函数时，称相应的算法为指数时间算法。

Growth Rate: sketch



复杂度与问题规模

算法复杂度	当前	计算机提速 100倍	计算机提速 10000倍
$O(n)$	n	$100n$	$10000n$
$O(n \log(n))$	n	$\sim 100n$	$\sim 10000n$
$O(n^2)$	n	$10n$	$100n$
$O(n^5)$	n	$2.5n$	$6.3n$
$O(2^n)$	n	$n + 6.64$	$n + 13.29$
$O(3^n)$	n	$n + 4.19$	$n + 8.38$

- 多项式时间算法的优点：
- (1) 随着问题输入规模的增加，算法的计算量（即算法复杂性）呈多项式增长；
- (2) 一个多项式时间算法利用另一个多项式时间算法作为其“子程序”，构造一个新的复合型算法，则新算法仍是多项式时间算法。

复杂度

- 指数型算法
 - 算法计算量是问题规模的指数函数
 - 只能够处理规模很小的问题
- 多项式型算法
 - 算法计算量是问题规模的多项式函数
 - 可以处理规模较大的问题

单纯形算法的复杂性

$$\begin{aligned} \max \quad & x_0 = \sum_{i=1}^n 10^{n-i} x_i \\ s.t. \quad & x_i + 2 \sum_{j<i} 10^{i-j} x_j \leq 10^{2i-2}, i = 2, \dots, n \\ & x_i \geq 0, \quad i = 1, 2, \dots, n \end{aligned}$$

二进制输入大小 $d(I) \leq 6n^2 + 2n + 3n + \log_2 |P|$,
其中 P 为所有非零系数的乘积。

单纯形算法需要 $2^n - 1$ 次迭代。

$$\max \quad \sum_{i=1}^n 2^{n-i} x_i$$

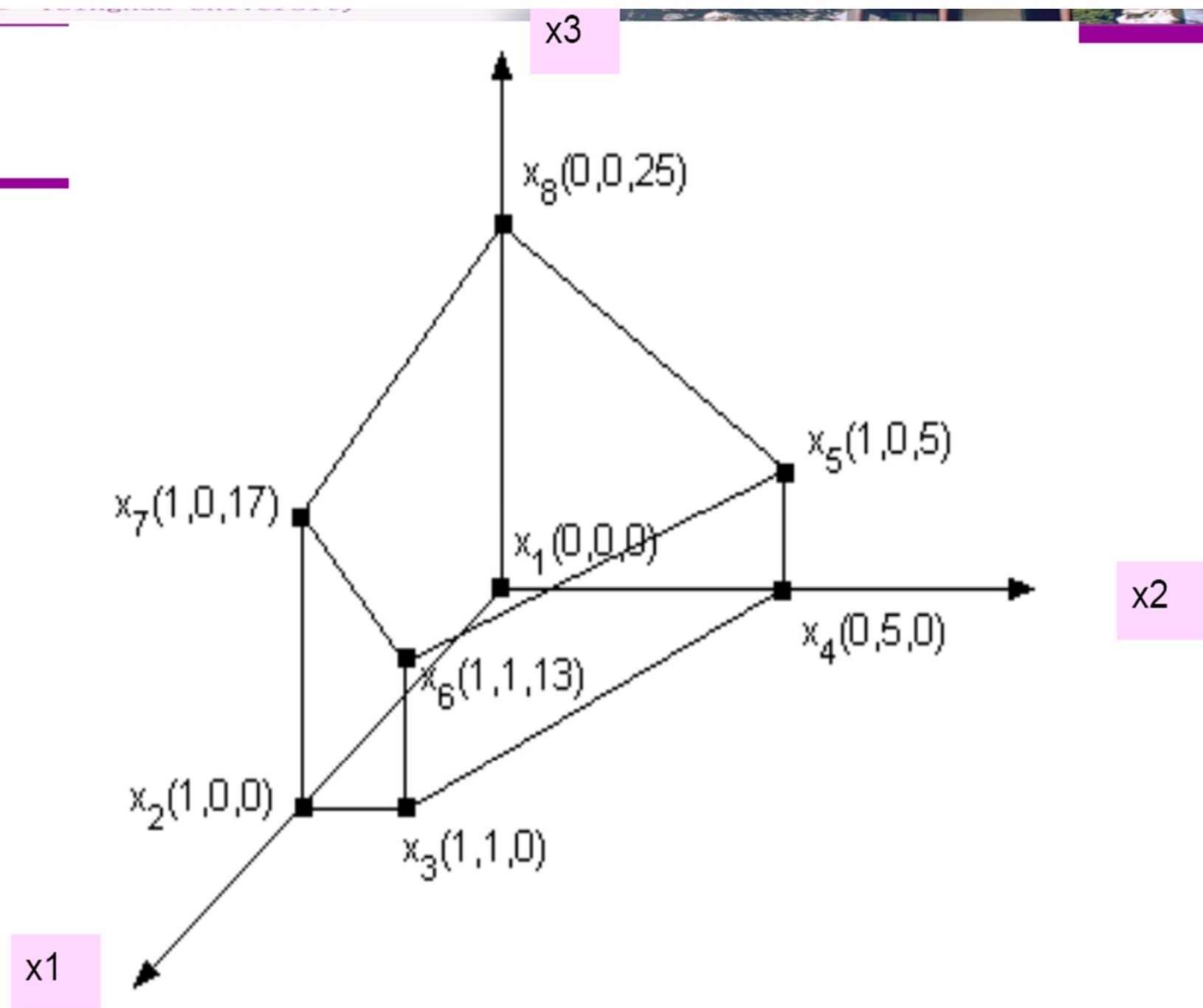
$$s.t. \quad x_i + 2 \sum_{j=1}^{i-1} 2^{i-j} x_j \leq a^{i-1}, \quad i = 2, \dots, n$$

$$x_i \geq 0, \quad i = 1, 2, \dots, n$$

定理：当 $a > 2$ 时，用单纯形算法求解上述问题时需要 $2^n - 1$ 次迭代。

Example: $n=3$, $a=5$

$$\begin{array}{ll}\max & 4x_1 + 2x_2 + x_3 \\s.t. & x_1 \leq 1 \\ & 4x_1 + x_2 \leq 5 \\ & 8x_1 + 4x_2 + x_3 \leq 25 \\ & x_1, x_2, x_3 \geq 0\end{array}$$

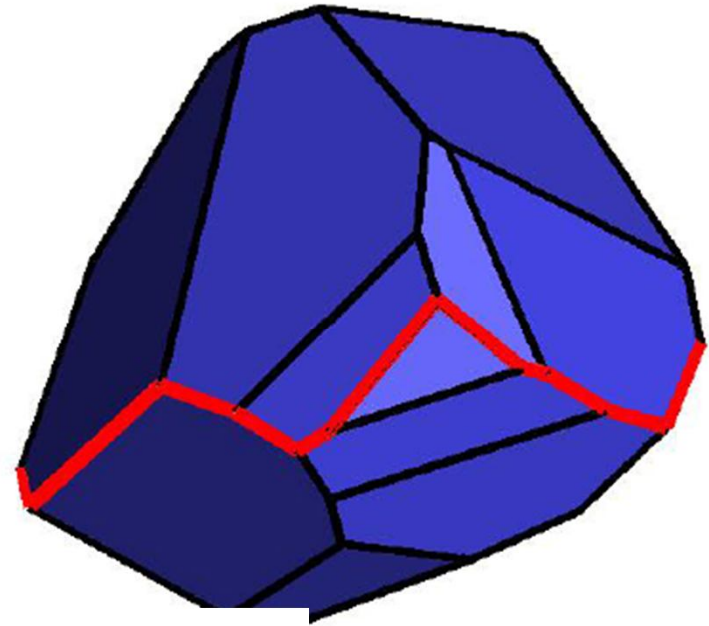


Simplex Method for Linear Programming

$$\min \quad cx$$

$$s.t. \quad Ax = b$$

$$x \geq 0$$



- Worst-Case: exponential
- Average-Case: polynomial
- Widely used in practice

History of Linear Programming

- Simplex Method (Dantzig, '47)
Exponential Worst-Case (Klee-Minty '72)
Avg-Case Analysis (Borgwardt '77, Smale '82,
Haimovich, Adler, Megiddo, Shamir, Karp, Todd)
- Ellipsoid Method (Khachiyan, '79) $O(n^4 L)$
- Interior-Point Method (Karmarkar, '84) $(n^{3.5} L)$

椭球法

第一个可以在多项式时间内解决一般线性规划问题的解法。

$$(P) \quad \begin{cases} \min cx \\ s.t. \quad Ax \geq b \\ x \geq 0 \end{cases} \quad (D) \quad \begin{cases} \max b^T w \\ s.t. \quad A^T w \leq c \\ w \geq 0 \end{cases}$$

根据(P) 与(D) 的对偶关系, 我们可将两者的最优解以一组最优性条件联结起来:

$$\begin{cases} Ax \geq b, & x \geq 0 \\ A^T w \leq c, & w \geq 0 \\ cx - wb = 0 \end{cases} \quad (*)$$

定理：存在求解 LP 问题的多项式时间算法的充要条件是存在求解线性不等式组 $Ax \leq b$ 的多项式时间算法。

证明：与线性不等式组 $Ax \leq b$ 相关的 LP 问题为

$$(*) \quad \begin{cases} \min cx \\ s.t. \quad \bar{A}\bar{x} \geq \bar{b} \\ \bar{x} \geq 0 \end{cases}$$

其中 $\bar{x} = \begin{pmatrix} x^+ \\ x^- \end{pmatrix}$, $\bar{A} = (-A \quad A)$, $\bar{b} = -b$, $x^+, x^- \geq 0$, c 任取(如 $c = 0$)

若有多项式时间的 LP 算法，能够判断问题 $(*)$ 不可行，则不等式组 $Ax \leq b$ 无解；或者得到其最优解或判定问题无界，则得到不等式组 $Ax \leq b$ 的一个解，显然就以多项式时间解决了问题 $Ax \leq b$ 。

定理：存在求解 LP 问题的多项式时间算法的充要条件是存在求解线性不等式组 $Ax \leq b$ 的多项式时间算法。

(*)的对偶问题为

$$\begin{cases} \max & wb \\ \text{s.t.} & w\bar{A} \leq c \\ & w \geq 0 \end{cases}$$

所以求解 LP 问题可归结为求解关于变量 (\bar{x}, w) 的

线性不等式组： $\bar{A}\bar{x} \geq \bar{b}, w\bar{A} \leq c, c\bar{x} - w\bar{b} \leq 0, \bar{x}, w \geq 0$

设有多项式时间方法求解线性不等式组。若该联立不等式组有解 (x^*, w^*) ，则 x^* 是 LP 问题的最优解， w^* 是其对偶问题的最优解；若该联立不等式组无解，考虑不等式组

$$\bar{A}\bar{x} \geq \bar{b}, \bar{x} \geq 0$$

若它有解，则 LP 问题无界；否则 LP 问题不可行。

只要能有效的解决最优性条件的线性不等式, 就能夠同时的解决一个线性规划问题(P) 以及它的对偶问题(D)。椭球法正是一种专门解决线性不等式的方法。

介绍如何以椭球法來解一组线性不等式 $Mu \leq v$

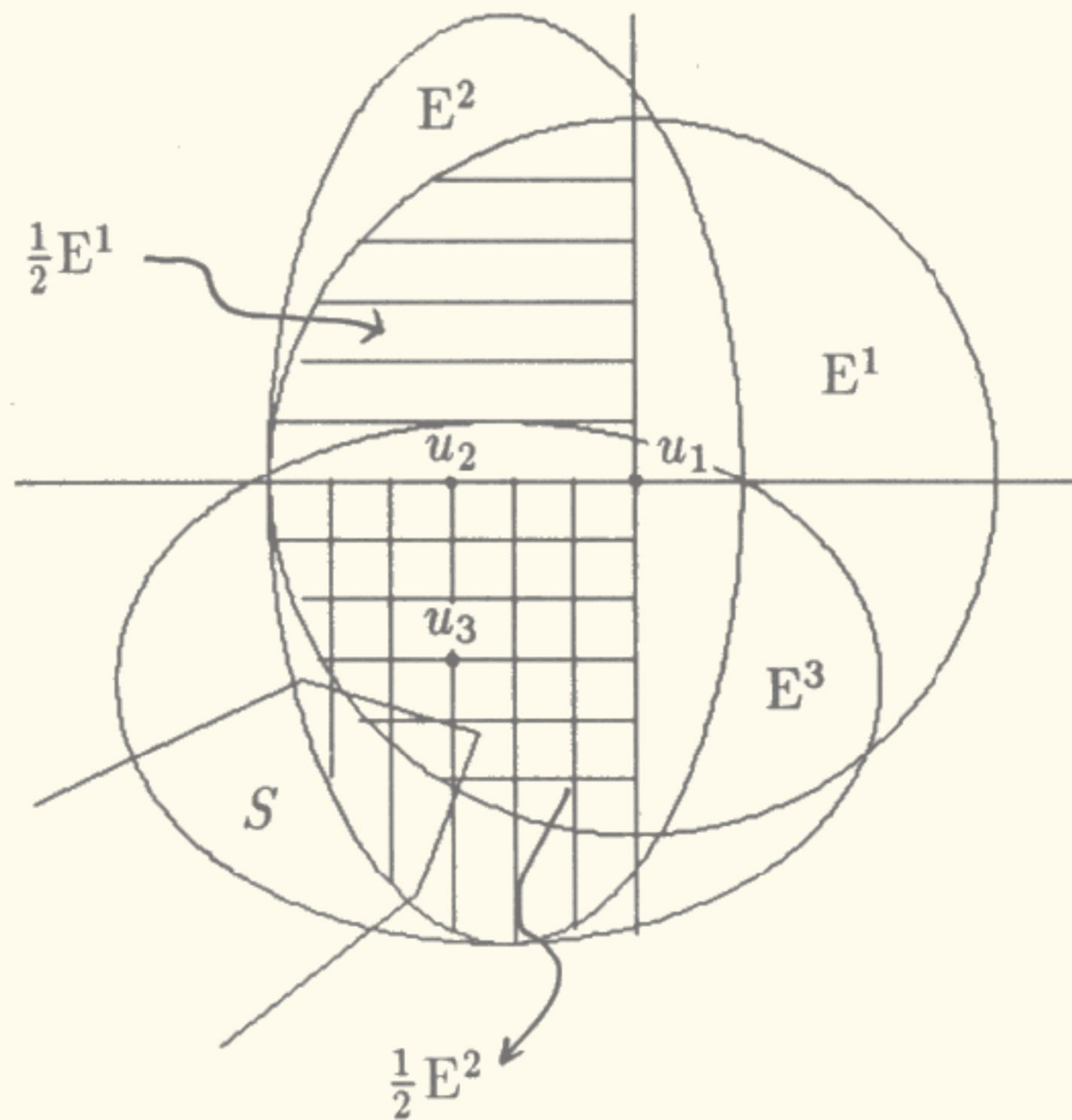
$Mu \leq v$ 的解集合是一个凸集: $S = \{u \mid Mu \leq v\}$

假设 $S \neq \emptyset$, 以原点 u^1 为圆心, 足夠大的半径做一圆 E^1 , 使得 $S \cap E^1 \neq \emptyset$ 。

若 $Mu^1 \leq v$, 则 u^1 为所求的解。

否则, 因为 $S \cap E^1$ 是一个凸集, 所以经过圆心, 可以切掉不含 $S \cap E^1$ 的半个圆, 而只剩下包含 $S \cap E^1$ 的半个圆(以 $1/2E^1$ 表示)。对 $1/2E^1$ 而言, 可以做出一个最小的椭圆 E^2 , 使得 $1/2E^1 \subseteq E^2$, 椭圆的圆心记 u^2 。

$S \cap E^1 \subseteq E^2$, 若 $Mu^2 \leq v$ 成立, 则 $u^2 \in S \cap E^1$ 必为其解。否则经过 u^2 又可切去半个 E^2 , 而使 $S \cap E^1$ 包含在另一半椭圆 $1/2E^2$ 之中。



在 p 维空间中, 每次做出的椭球体积都会逐渐缩小。以 $V(E^k)$ 及 $V(E^{k+1})$ 来表示前后两个椭球的体积, 那么可以证明 $V(E^{k+1}) < e^{-1/2(p+1)} V(E^k)$ 。所以

$$V(E^{k+1}) < e^{-k/2(p+1)} V(E^1)。$$

表明在多项式时间内, 新的椭球体积便可缩减至零, 否则原题无解。

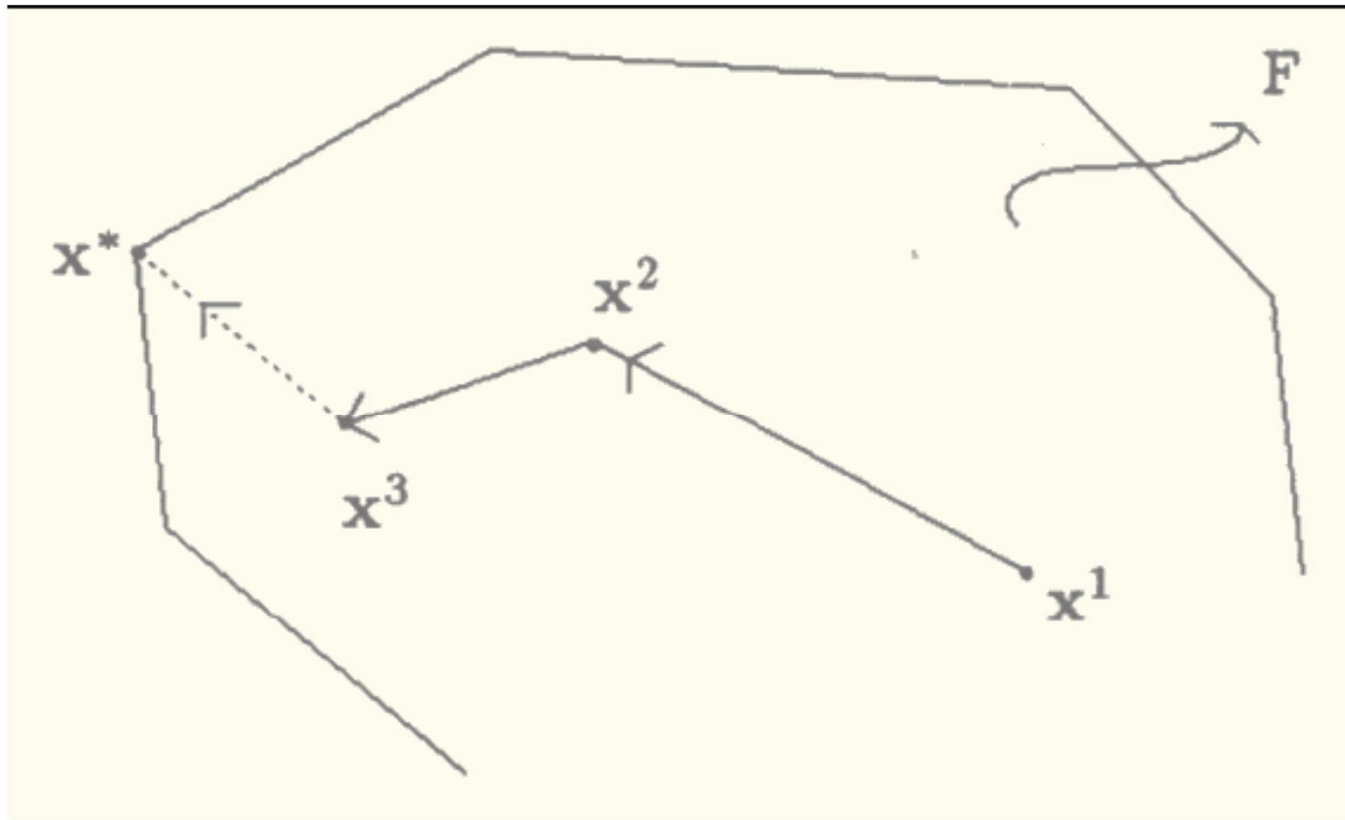
步骤:

1: 考虑最优化条件的线性不等式 $(*)$ 。在 $(x, w) \in R^{n+m}$ 的空间中, 以原点 $u^1=(x^1, w^1)$ 为心, 足够大的正数 2^{2L} 为半径做一圆球 E^1 。置 $k=1$ 。

2: 检验现有球心 $u^k=(x^k, w^k)$ 是否满足该最优化不等式。若满足, 则 x^k 是(P)的最优解, w^k 是(D)的最优解。若不满足, 则依前述方法得到一个新的椭球 E^{k+1} 及其球心 u^{k+1} 。

3: 若是 $V(E^{k+1}) = 0$, u^{k+1} 仍不满足最优化不等式, 则(P)或(D)至少有一无最优解。否则置 $k := k + 1$, 返回**2**。

内点法



$$(P) \quad \begin{cases} \min c^T x \\ s.t. \quad Ax = b \\ x \geq 0 \end{cases}$$

称 $x \in R^n$ 为一个可行内点 (**interior feasible**)
如果 $Ax = b, x > 0$ 。

可行内点解集合: $F^0 = \{x \in R^n | Ax = b, x > 0\}$ 。

假设 F^0 非空。

内点法可粗略的分为三个步骤:

步骤一: 找一个可行内点 $x^1 \in F^0$ 。置 $k=1$ 。

步骤二: 决定现有解 x^k 是否为(P)的最优解。若是, 则输出 $x^* = x^k$ 。否则就寻找一个好的移动方向 d_x^k , 以及适当的步长 $\alpha_k > 0$ 。

步骤三: 由 x^k 移动到新的内点

$$x^{k+1} = x^k + \alpha_k d_x^k \in F^0.$$

置 $k := k + 1$, 返回步骤二。

Primal Affine Scaling内点法

假想 (P) 的可行解集合位于第一象限的一个球, 而现行解 x^k 落在球心上, 此球的半径是 $r > 0$ 。

$$(P_1) \quad \begin{cases} \min c^T x \\ s.t. \quad \sum_{j=1}^n (x_j - x_j^k)^2 \leq r^2 \end{cases}$$

最优解为:

$$x^* = x^{k+1} = x^k + r \frac{-c}{\| -c \|}$$

将 (P_1) 变得稍微复杂一些, 将球改为第一象限来考虑下列问题:

$$(P_2) \quad \begin{cases} \min c^T x \\ s.t. \quad x \geq 0 \end{cases}$$

假设 $x^k > 0$, 定义 $n \times n$ 矩阵

$$D_k = \begin{bmatrix} x_1^k & & & 0 \\ & x_2^k & & \\ & & \ddots & \\ 0 & & & x_n^k \end{bmatrix} \quad D_k^{-1} = \begin{bmatrix} \frac{1}{x_1^k} & & & 0 \\ & \frac{1}{x_2^k} & & \\ & & \ddots & \\ 0 & & & \frac{1}{x_n^k} \end{bmatrix}$$

定义映射:

$$T_k : R_+^n \rightarrow R_+^n$$

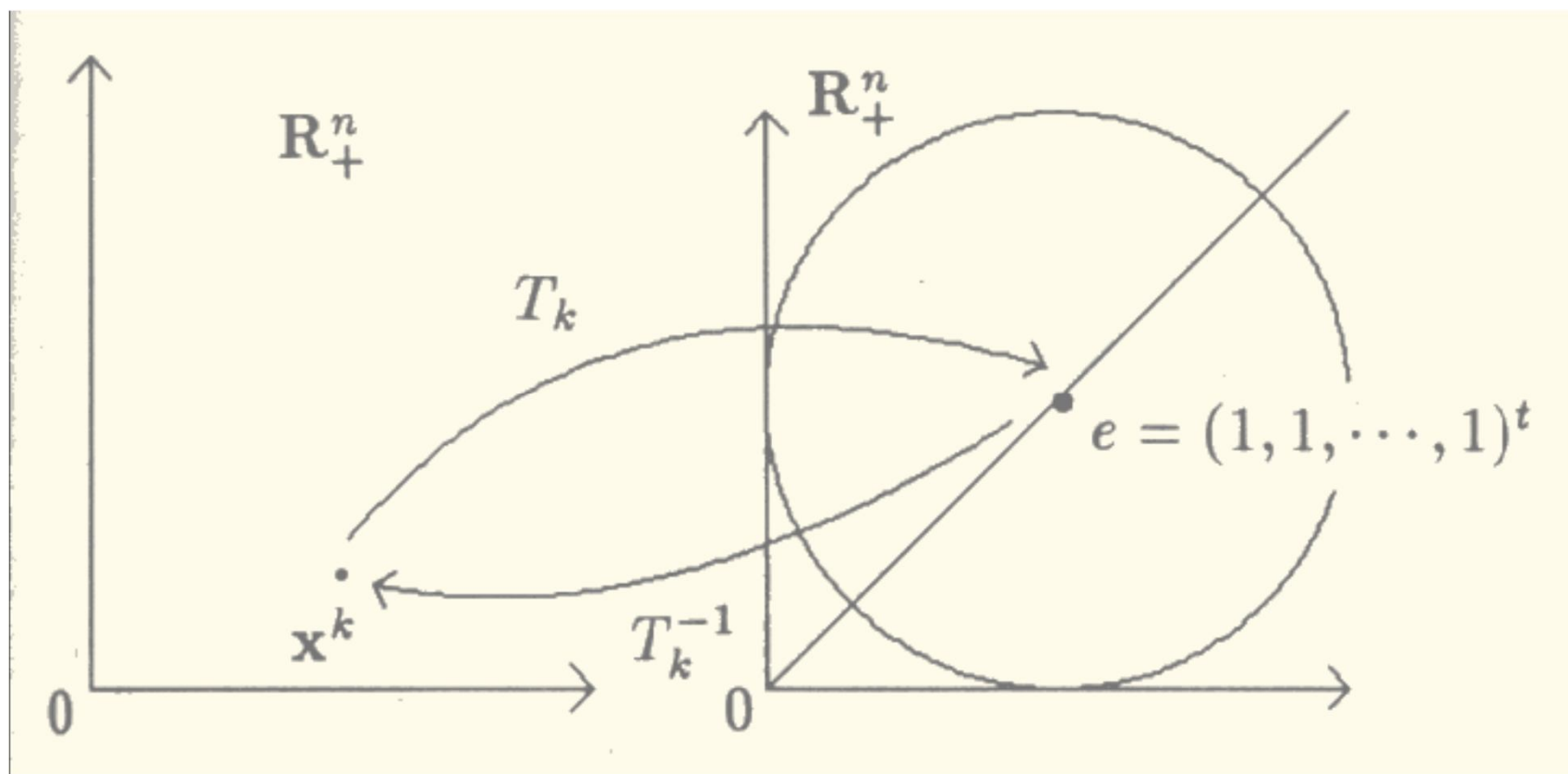
$$x \rightarrow y = T_k(x) \stackrel{\Delta}{=} D_k^{-1} x \quad \text{则}$$

$$T_k^{-1} : R_+^n \rightarrow R_+^n$$

$$y \rightarrow x = T_k^{-1}(y) = D_k y$$

$$y^k = T_k(x^k) = D_k^{-1} x^k = e$$

$$T_k^{-1}(e) = D_k e = x^k$$



在上述转化之下, (P_2) 的近似问题在 y 空间中就可写成:

$$(P'_2) \quad \begin{cases} \min c^T D_k y \\ s.t. \quad \sum_{j=1}^n (y_j - 1)^2 \leq 1 \end{cases}$$

应用 (P_1) 的解答技巧, 在 y 空间中的近似最优解为:

$$y^* = y^{k+1} = y^k + \frac{-D_k c}{\|D_k c\|} = e - \frac{D_k c}{\|D_k c\|}$$

在 x 空间中 (P_2) 的近似最优解为:

$$x^* = x^{k+1} = D_k y^{k+1} = D_k e - \frac{D_k^2 c}{\|D_k c\|} = x^k - \frac{D_k^2 c}{\|D_k c\|}$$

$$(P) \quad \begin{cases} \min c^T x \\ s.t. \quad Ax = b \\ x \geq 0 \end{cases}$$

在 y 空間中**(P)** 变成下列的近似问题:

$$(P') \quad \begin{cases} \min c^T D_k y \\ s.t. \quad AD_k y = b \\ \sum_{j=1}^n (y_j - 1)^2 \leq 1 \end{cases}$$

与 (P'_2) 相比较, (P') 多了一些等式约束条件 $AD_k y = b$ 。为了保持这些等式的继续成立, 原来在 (P'_2) 中的移动方向 $-D_k c$ 便需要投影在 (AD_k) 这个矩阵的零空间(null space) 之中, 即经过投影后的方向应是 $P_k(-D_k c)$, 其中 P_k 是一个投影矩阵, 定义为:

$$P_k = \left[I - D_k A^T \left(AD_k^2 A^T \right)^{-1} AD_k \right]$$

所以 (P') 的近似最优解是 $y^* = y^{k+1} = e + \frac{P_k(-D_k c)}{\|P_k(-D_k c)\|}$

(P) 的近似最优解是 $x^* = x^{k+1} = D_k y^{k+1} = x^k - \frac{D_k P_k D_k c}{\|P_k D_k c\|}$

定理：若存在 $j, j = 1, 2, \dots, n$ 使得 $x_j^{k+1} = 0$ ，则 x^{k+1} 是 (P) 的最优解。

Primal Affine Scaling内点法的重要步骤:

1. 找一个可行内点 $x^1 \in F^0$ ，置 $k := 1$ 。

2. 计算移动方向 $d_x^k = -D_k P_k D_k c$ ，步长 $\alpha_k = \frac{1}{\|P_k D_k c\|}$ ，

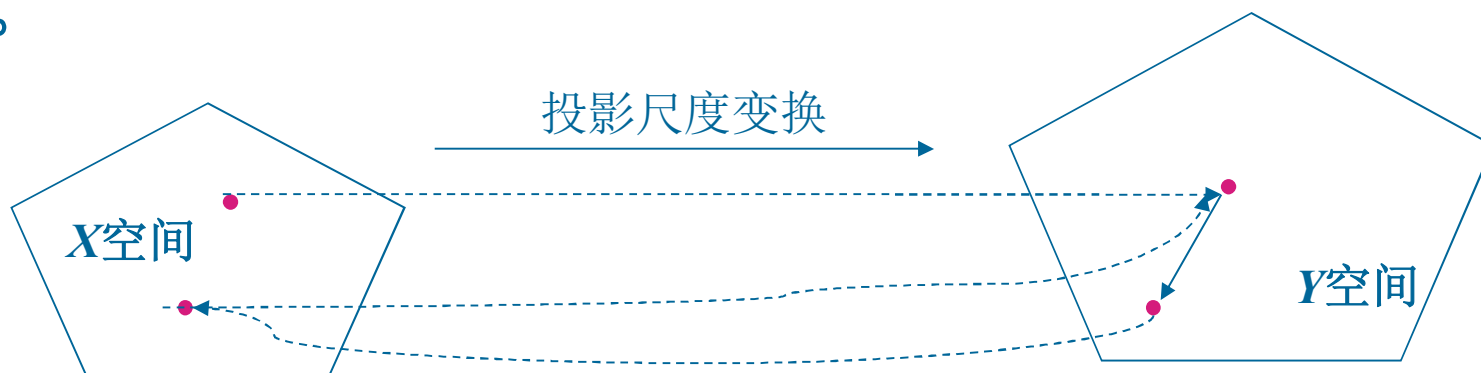
新内点 $x^{k+1} = x^k + \alpha_k d_x^k$ 。

3. 检验是否有 $x_j^{k+1} = 0, j = 1, 2, \dots, n$ ，若成立，则输出 $x^* = x^{k+1}$ ；否则置 $k := k + 1$ ，返回2。

Karmarkar投影尺度算法

- 两个基本事实：
- **(1)** 如果一个内点居于多面体的中心，那么目标函数的负梯度方向是比较好的可行方向；
- **(2)** 在不改变问题基本特性的条件下，存在一个适当的变换，能够将可行域中给定的内点置于变换后的可行域的中心。

- 基本思想：
- **(1)** 选定一个内点解作为迭代过程的初始点，利用可行域的投影尺度变换，将当前的内点解置于变换后的可行域的中心；
- **(2)** 在变换后的可行域中沿着目标函数最速下降方向的正交投影移动，获得新的可行内点，并通过投影尺度逆变换将新的可行内点映射到原来的可行域，作为新的迭代点。
- **(3)** 重复这一过程，直至求出满足一定精度的近优解。



Karmarkar标准形

$$(*) \quad \begin{cases} \min & c^T x \\ s.t. & Ax = 0 \\ & x_1 + x_2 + \cdots + x_n = 1 \\ & x \geq 0 \end{cases}$$

其中 A 是 $m \times n$ 满秩矩阵, $c, x \in R^n$.

基本假设: (1) 问题 $(*)$ 是可行的, 单纯形 $S = \left\{ x \mid \sum_{j=1}^n x_j = 1, x \geq 0 \right\}$

的中心 $a^{(0)} = \frac{1}{n}e$ 是可行点, 其中 $e = (1, \dots, 1)^T$.

(2) 对每个可行点 x , 有 $c^T x \geq 0$ 。

(3) 终止参数 q 给定, 目的是求一个可行点 x , 使得

$$\frac{c^T x}{c^T a^{(0)}} \leq 2^{-q}.$$

单纯形

定义集合 $S = \{x \mid e^T x = 1, x \geq 0\}$ 为 $n-1$ 维单纯形,

其顶点是 $e_j = (0, \dots, 0, 1, 0, \dots, 0)^T, j = 1, \dots, n$.

S 的中心是 $a^{(0)} = \frac{1}{n}e$.

单纯形 S 的外接球半径

$$R = \sqrt{\left(1 - \frac{1}{n}\right)^2 + (n-1)\left(0 - \frac{1}{n}\right)^2} = \sqrt{\frac{n-1}{n}}.$$

单纯形 S 的内接球半径

$$r = \sqrt{\left(0 - \frac{1}{n}\right)^2 + (n-1)\left(\frac{1}{n} - \frac{1}{n-1}\right)^2} = \frac{1}{\sqrt{n(n-1)}}.$$

外接球与内接球半径之比 $R/r = n-1$.

向量的投影

$B_{m \times n}$, $r(B) = m$. 它的 m 个行向量生成的子空间为 V_B .

V_B^\perp : V_B 的正交子空间, 则 $V_B^\perp = \{x \mid Bx = 0\}$,

则 $R^n = V_B + V_B^\perp$.

$\forall y \in R^n$, $y = y_1 + y_2$, 其中 $y_1 \in V_B$, $y_2 \in V_B^\perp$, 则 $y_1 = B^T \lambda$

$$\Rightarrow y = B^T \lambda + y_2$$

两边左乘 B , 得 $By = BB^T \lambda$

$$\because r(B) = m, \therefore \lambda = (BB^T)^{-1} By$$

$$\Rightarrow y_1 = B^T (BB^T)^{-1} By$$

$$y_2 = \left[I - B^T (BB^T)^{-1} B \right] y.$$

正交投影
矩阵

单纯形 S 的投影尺度变换

定义 $T(x) = \frac{D^{-1}x}{e^T D^{-1}x},$

其中 $D = \text{diag}(a_1, \dots, a_n), a_i > 0, i = 1, \dots, n.$

记 $T(x) = y$, 则

$$y_i = \frac{x_i/a_i}{\sum_j (x_j/a_j)}, i = 1, \dots, n.$$

$$\sum_{i=1}^n y_i = 1$$

变换 T 的性质

1. T 是单纯形 S 上的一一对应，其逆变换为

$$x = T^{-1}(y) = \frac{Dy}{e^T Dy} \quad (*) \quad \left(x_i = \frac{a_i y_i}{\sum_j a_j y_j} \right).$$

S 的每个顶点 $e_j = (0, \dots, 0, 1, 0, \dots, 0)^T$ 的像还是这个顶点。

2. 约束 $Ax = 0$ 在变换 $(*)$ 下等价于 $ADy = 0$.

3. 点 $a = (a_1, \dots, a_n)^T$ 的像是单纯形 S 的中心 $\frac{1}{n}e = (a^{(0)})$.

4. T 把由 $x_j = 0$ 给出的 S 的每个面映射成对应的面 $y_j = 0$.

5. 若 $a \in \{x \mid Ax = 0\}$ ，则 S 的中心 $a^{(0)} \in \{y \mid ADy = 0\}$.

势函数

对每个线性函数 $l(x)$ ，定义与它相联系的势函数为：

$$f(x) = \sum_{j=1}^n \ln \frac{l(x)}{x_j} + k \quad (\text{其中} k \text{为常数})$$

性质：

- 1 . 射影变换 T 把势函数变换成具有相同形式的函数.
- 2 . 目标函数值所期望的下降量可通过势函数值的充分小来达到.
- 3 . 优化函数 $f(x)$ 可用优化线性函数来近似.

$$\text{令 } l(x) = c^T x$$

$$x = \frac{Dy}{e^T Dy} \quad x_i = \frac{a_i y_i}{\sum_j a_j y_j} = \frac{a_i y_i}{e^T Dy}$$

$$\begin{aligned} \therefore f(x) &= \sum_{j=1}^n \ln \frac{c^T x}{x_j} = \sum_{j=1}^n \ln \frac{c^T Dy / (e^T Dy)}{a_j y_j / (e^T Dy)} \\ &= \sum_{j=1}^n \ln \frac{c^T Dy}{a_j y_j} = \sum_{j=1}^n \ln \frac{c'^T y}{y_j} - \sum_{j=1}^n \ln a_j \end{aligned}$$

即变换后的势函数为：

$$\tilde{f}(y) = \sum_{j=1}^n \ln \frac{c'^T y}{y_j} - \sum_{j=1}^n \ln a_j$$

其中 $c' = Dc$.

求解方法

$$(*) \quad \begin{cases} \min & c^T x \\ s.t. & Ax = 0 \\ & x_1 + x_2 + \cdots + x_n = 1 \\ & x \geq 0 \end{cases}$$

其中 A 是 $m \times n$ 满秩矩阵, $c, x \in R^n$.

$$(*) \quad \begin{cases} \min & c^T x \\ s.t. & x \in S \cap \Omega \end{cases}$$

其中 $\Omega = \{x \mid Ax = 0\}$, $S = \left\{x \mid \sum_{j=1}^n x_j = 1, x \geq 0\right\}$

运用变换 $x = T^{-1}(y) = \frac{Dy}{e^T Dy}$ 把 Ω 变为 $\Omega' = \{y \mid ADy = 0\}$,

S 变为 S , 即把可行域 $S \cap \Omega$ 变为 $S \cap \Omega'$, 同时把单纯形上

的可行点 $a = (a_1, \dots, a_n)^T > 0$ 变成单纯形 S 的中心 $a^{(0)} = \frac{1}{n}e$,

则 $a^{(0)} \in S \cap \Omega'$ 。

$$c^T x \quad \Rightarrow \quad \frac{c^T Dy}{e^T Dy}$$

$$f(x) = \sum_{j=1}^n \ln \frac{c^T x}{x_j} \quad \Rightarrow \quad \tilde{f}(y) = \sum_{j=1}^n \ln \frac{c'^T y}{y_j} - \sum_{j=1}^n \ln a_j$$

$$\begin{cases} \min & c^T x \\ s.t. & x \in S \cap \Omega \end{cases} \Rightarrow \begin{cases} \min & \tilde{f}(y) \\ s.t. & y \in S \cap \Omega' \end{cases}$$

用包含在 S 内、以 $a^{(0)}$ 为球心， αr 为半径的球

$B(a^{(0)}, \alpha r)$ 取代 S ，其中 $r = \frac{1}{\sqrt{n(n-1)}}$ ， $\alpha \in (0,1)$ 。记

$$B'(a^{(0)}, \alpha r) = B(a^{(0)}, \alpha r) \cap \Omega'$$

则对 $\forall y \in B'(a^{(0)}, \alpha r)$ ，有

$$ADy = 0 \quad \text{且}$$

$$\sum_{j=1}^n y_j = 1, \quad \text{即 } e^T y = 1.$$

$$\text{令 } B = \begin{bmatrix} AD \\ e^T \end{bmatrix} \quad e_{m+1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{array}{l} ADy = 0 \\ e^T y = 1 \end{array} \quad \longleftrightarrow \quad By = e_{m+1}$$

$$\text{令 } \Omega'' = \{y \mid By = e_{m+1}\}$$

$$\begin{cases} \min & \tilde{f}(y) \\ \text{s.t.} & y \in S \cap \Omega' \end{cases} \Rightarrow \begin{cases} \min & \tilde{f}(y) \\ \text{s.t.} & y \in B(a^{(0)}, \alpha r) \cap \Omega'' \end{cases}$$

$$\begin{cases} \min & \tilde{f}(y) \\ \text{s.t.} & y \in B(a^{(0)}, \alpha r) \cap \Omega'' \end{cases} \Rightarrow \begin{cases} \min & c'^T y \\ \text{s.t.} & y \in B(a^{(0)}, \alpha r) \cap \Omega'' \end{cases}$$

其中 $c' = Dc$ 。

记 $c_p = \left(I - B^T (BB^T)^{-1} B \right) Dc$

$c'^T y$ 在 Ω'' 上下降最快的方向是

$$d^{(0)} = -\frac{c_p}{\|c_p\|}$$

$c'^T y$ 在 $B(a^{(0)}, \alpha r)$ 上的最小点是

$$b' = a^{(0)} - \frac{\alpha r}{\|c_p\|} c_p$$

Karmarkar投影尺度算法

1. 置 $k = 0$, $x^{(0)} = \frac{1}{n}e$, $r = \sqrt{1/(n-1)n}$; 参数 $\alpha \in (0,1)$, L 是充分大的正整数.

2. 若 $c^T x^{(k)} \leq 2^{-L} c^T x^{(0)}$, 则停止计算, 得到近优解 $x^{(k)}$; 否则转3。

$$3. \text{ 令 } X^{(k)} = \text{diag}(x^{(k)}), \quad B_k = \begin{pmatrix} AX^{(k)} \\ e^T \end{pmatrix},$$

$$d_y^k = -\left(I - B_k^T (B_k B_k^T)^{-1} B_k\right) X^{(k)} c,$$

$$y^{k+1} = \frac{1}{n}e + \alpha r \frac{d_y^k}{\|d_y^k\|}, \quad x^{(k+1)} = \frac{X^{(k)} y^{k+1}}{e^T X^{(k)} y^{k+1}}.$$

4. 计算势函数值 $f(x^{(k)})$ 和 $f(x^{(k+1)})$, 若 $f(x^{(k)}) - f(x^{(k+1)}) < \delta$, 则停止计算, 得出 *Karmarkar* 标准问题的极小值大于0的结论, 这种情形对应原来的线性规划问题不可行或无下界; 若 $f(x^{(k)}) - f(x^{(k+1)}) \geq 0$, 则转5。

5. 置 $k := k + 1$, 返回2。

有关定理

定理1. 算法在 $O(n(q + \log n))$ 步内求得一个可行点 x , 使得

$$\frac{c^T x}{c^T a^{(0)}} \leq 2^{-q}.$$

定理2. 对每个 k , 有下列两种情形之一:

(1) $f(x^{(k+1)}) \leq f(x^{(k)}) - \delta;$

(2) 目标函数的极小值大于0。其中

$$\delta = \alpha - \frac{\alpha^2}{2} - \frac{\alpha^2 n}{(n-1)[1 - \alpha \sqrt{n/(n-1)}]}.$$

定理3. 存在点 $b' \in B(a^{(0)}, \alpha r) \cap \Omega''$, 使得

$$\tilde{f}(b') \leq \tilde{f}(a^{(0)}) - \delta.$$

定理4. 设 b' 是 $c'^T x$ 在 $B(a^{(0)}, \alpha r) \cap \Omega''$ 上的极小点,

则 $f'(b') \leq f'(a^{(0)}) - \delta.$

定理5. 算法求得的点 b' 是 $c'^T x$ 在 $B(a^{(0)}, \alpha r) \cap \Omega''$ 上的极小点.

一般线性规划问题的处理

1. 利用对偶定理转化线性规划问题

$$(P) \quad \begin{cases} \min c^T x \\ s.t. \quad Ax \geq b \\ \quad \quad x \geq 0 \end{cases} \quad (D) \quad \begin{cases} \max b^T u \\ s.t. \quad A^T u \leq c \\ \quad \quad u \geq 0 \end{cases}$$

(P)存在最优解的充要条件是

$$\left\{ \begin{array}{l} Ax \geq b \\ A^T u \leq c \\ c^T x - b^T u = 0 \\ x \geq 0 \\ u \geq 0 \end{array} \right. \longleftrightarrow \left\{ \begin{array}{l} Ax - y = b \\ A^T u + v = c \\ c^T x - b^T u = 0 \\ x \geq 0, u \geq 0, y \geq 0, v \geq 0 \end{array} \right.$$

记

$$\bar{A} = \begin{pmatrix} A & 0 & -I & 0 \\ 0 & A^T & 0 & I \\ c^T & -b^T & 0 & 0 \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} b \\ c \\ 0 \end{pmatrix}, \quad \bar{x} = \begin{pmatrix} x \\ u \\ y \\ v \end{pmatrix}$$

于是问题归结为求解：

$$\bar{A}\bar{x} = \bar{b}, \quad \bar{x} \geq 0$$

引入人工变量 λ ，上述问题又可重新化为一个线性规划问题：

$$\min \lambda$$

$$s.t. \quad \bar{A}\bar{x} - \lambda(\bar{A}e - \bar{b}) = \bar{b}$$

$$\bar{x} \geq 0, \lambda \geq 0$$

记

$$B = (\bar{A}, -(\bar{A}e - \bar{b})), \quad z = \begin{bmatrix} \bar{x} \\ \lambda \end{bmatrix}, \quad \bar{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\min \lambda$$

$$s.t. \quad \bar{A}\bar{x} - \lambda(\bar{A}e - \bar{b}) = \bar{b}$$
$$\bar{x} \geq 0, \lambda \geq 0$$

$$\min \bar{c}^T z$$

$$s.t. \quad Bz = \bar{b}$$
$$z \geq 0$$

两个特性:

1. $z = e$ 是问题的一个可行内解。
2. 若 $\min \bar{c}^T z > 0$, 则原线性规划问题无可行解或者无下界; 若 $\min \bar{c}^T z = 0$, 则同时也就获得了原线性规划问题和对偶问题的一个最优解。

进行如下投影变换：

$$\bar{z}_{[k]} = \frac{z}{e^T z + 1}, \quad \bar{z}_{k+1} = \frac{1}{e^T z + 1}$$

记向量 $\bar{y} = \left(\bar{z}_{[k]}^T, \bar{z}_{k+1} \right)^T$

$$\min \bar{c}^T z$$

$$s.t. \quad Bz = \bar{b}$$

$$z \geq 0$$



$$\min \left(\bar{c}^T, 0 \right) \bar{y}$$

$$s.t. \quad \left(B, -\bar{b} \right) \bar{y} = 0$$

$$e^T \bar{y} = 1$$

$$\bar{y} \geq 0$$

两个特性:

1. $\bar{y} = \frac{e}{k+1}$ 是问题的一个可行内点解。
2. 若最优值 = 0, 则同时也就获得了原线性规划问题和对偶问题的一个最优解。
3. 若最优值 > 0, 则原线性规划问题无可行解或无下界; 此时, 可以从 $\{x \mid Ax = b, x \geq 0\}$ 出发, 构造相应的 *Karmarkar* 标准形, 通过求解这个 *Karmarkar* 标准形, 即可判断原来的线性规划问题是无界还是无可行解。

三种解法比较

单纯形法最老牌,研究的最为透彻,商业化的软件程序也最成熟.

椭球法像昙花一现,虽然在理论上证明了线性规划问题可在多项式时间内求解,但在实际应用上反而不如单纯形法来得有效便捷。

内点法是最新的设计,理论上它比椭球法还要有效,实际应用上它也可与单纯形法相抗衡,不少的商业化软件已经上市,前景甚佳。