程序设计实习（实验班-2024春）稀疏性、压缩感知与线性规划方法

授课教师：姜少峰

助教：冯施源 吴天意

Email: shaofeng.jiang@pku.edu.cn

稀疏性以及如何利用稀疏性

- 稀疏数据，例如非0/近似非0元素较少的数据，在现实中很多见
- 高维数据经常是稀疏的，这就使得Curse of Dimensionality在现实中经常不成立
- 因此有效处理稀疏数据是应对Curse of Dimensionality的重要途径

传统的利用稀疏性的框架

- 一般而言，比较自然的利用稀疏性的方法是先把数据读进来，然后
  - 利用可以处理稀疏数据的算法进行计算，例如我们的线性回归/PCA的算法
  - 利用SVD等方法进行low-rank表示
  - …

压缩感知

Compressive Sensing

- 上述利用稀疏性的方法都是先拿到原始数据，然后再利用稀疏性进行降维/压缩
  - 但：数据本身的不少噪声/冗余本来可以不用理会的，也必须先读进来
- 可否不把数据全读进来，而是仅仅读/处理“需要的部分”?
- “压缩感知”即落实这种思想，将数据从一种“压缩”的形式中提取出来

压缩感知具体设定

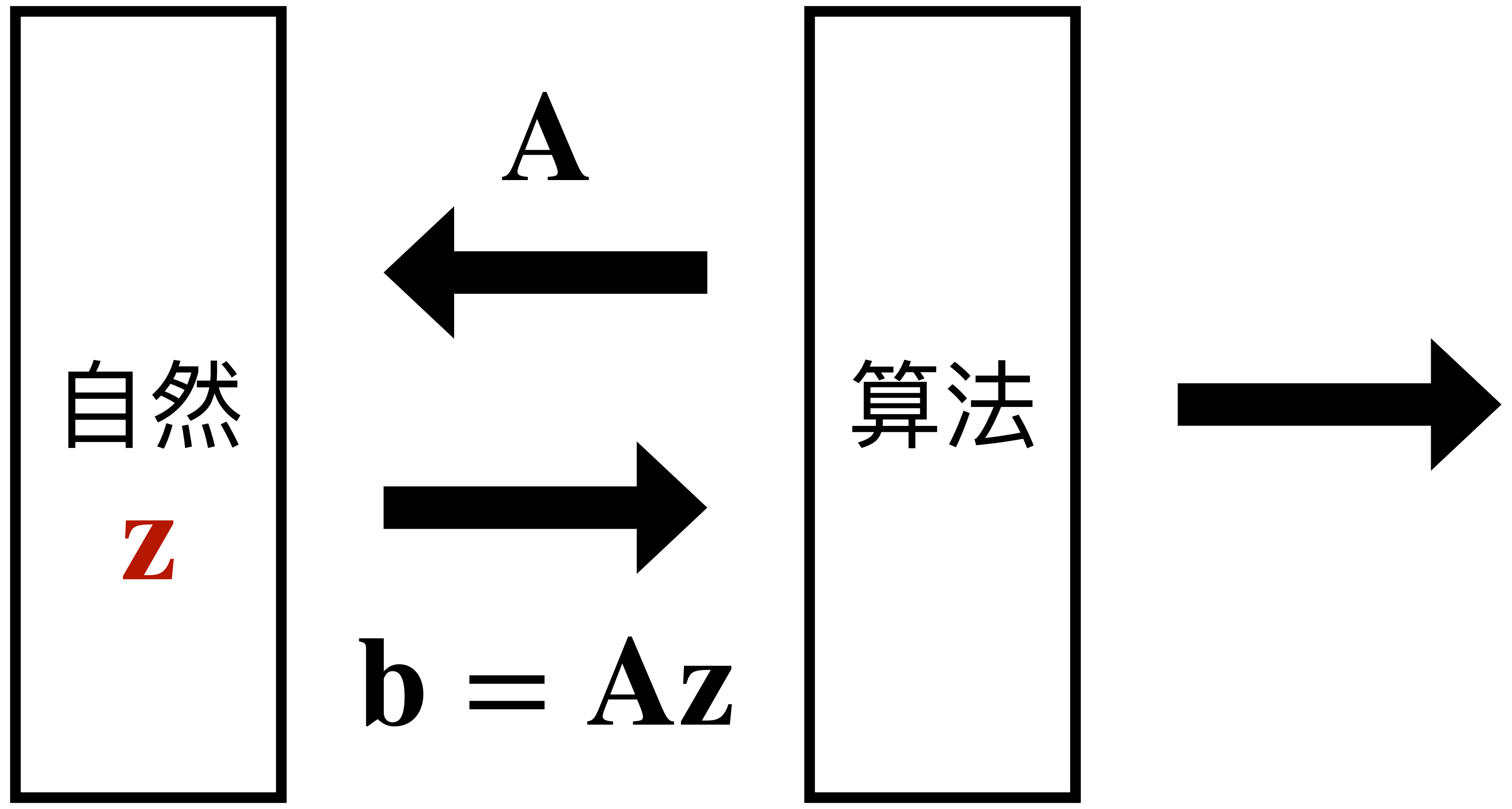
- 我们设计m个linear measurements（线性测量） $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{R}^n$
- “自然”生成一个未知的信号$\mathbf{z} \in \mathbb{R}^n$

目标是要同一组测量$\mathbf{a}_1, \ldots, \mathbf{a}_m$，对“所有”可能的$\mathbf{z}$都要有效恢复

- 某种硬件告诉我们信号在线性测量上的值$\mathbf{b}$，$b_1 = \langle \mathbf{a}_1, \mathbf{z} \rangle, \ldots, b_m = \langle \mathbf{a}_m, \mathbf{z} \rangle$

想象设计一个影像传感器（摄像头）
z就是真实图像，b就是传感器给出的测量值

- 仅从$\mathbf{b}$恢复$\mathbf{z}$



算法的输出恢复了$\mathbf{z}$

建模成矩阵

设$\mathbf{A} = \begin{bmatrix} -\mathbf{a}_1- \\ -\mathbf{a}_2- \\ \vdots \\ -\mathbf{a}_m- \end{bmatrix}$，硬件给出的观测是$\mathbf{b} \in \mathbf{R}^m$，满足$\mathbf{b} = \mathbf{A}\mathbf{z}$

如何设计$\mathbf{A}$来恢复出$\mathbf{z}$?

等价于解线性方程（组）

- 注意到问题等价于求解$\mathbf{x}$，使得$\mathbf{A}\mathbf{x} = \mathbf{b}$
- 首先，$\mathbf{x} = \mathbf{z}$一定是这个方程的解，因为观测$\mathbf{b}$保证有$\mathbf{A}\mathbf{z} = \mathbf{b}$
- 因此，如果$m = n$应该就能精确解出$\mathbf{z}$

$\mathbf{z}$有n维，因此n个方程n个未知数有唯一解
此时可以取$\mathbf{A} = \mathbf{I}_n$，即全1对角阵

但$m < n$会怎样?

- 一般来说如果$m < n$，则$\mathbf{Ax} = \mathbf{b}$为一个欠定方程
    - $\mathbf{z}$仍然是解，但是解不唯一！
- 这说明若$\mathbf{z}$可以取任意变量，那么对于恢复$\mathbf{z}$来说，$m = n$是必要的（也充分）
- 但是如果$\mathbf{z}$是稀疏的呢？是否用$m \ll n$就可以找到$\mathbf{z}$呢?

z的稀疏性

- 我们的设定：设z是k-稀疏的，即z有至多k个非0坐标，并且一般假定$k \ll n$
- 例如，z可以是一幅黑白图像，纯黑图像中有少数白线

  纯黑值 = 0

  白线 = 1，但总共只有$k = O(\sqrt{n})$个像素

- 我们只是考虑在默认坐标下z有k个非0，但有时需要考虑先进行某些变换
  - 例如图像可能是灰度的，那么需要先二值化
  - 可能需要变换坐标，之后才会稀疏

压缩感知定理

提出该定理的论文可参考[Donoho, IEEE Trans. Inf 2006]
和[Candes-Romberg-Tao, IEEE Trans. Inf 2006]

定理：设$\mathbf{A} \in \mathbb{R}^{m\times n}$其中$m = O(k\log(n/k))$且$\mathbf{A}$的每个元素都从$\mathcal{N}(0,1)$独立采样。则$\mathbf{A}$以高概率，同时对所有的$n$维的$k$-稀疏的$\mathbf{z}$，“可以”从观测$\mathbf{b} = \mathbf{A}\mathbf{z}$中精确恢复$\mathbf{z}$

即使对于1维，也有无穷多种可能性
这里强调同一个$\mathbf{A}$可对所有这些无穷种情况同时奏效

这个版本暂时没有告知如何才能恢复，只是告知了$\mathbf{A}$如何选取

定理证明不在本课程范畴；我们讨论如何理解、应用

可以扩展到近似稀疏性吗?

- 该定理的保证其实也对“近似”k-稀疏的$\mathbf{z}$“成立”
  - 近似k-稀疏：$\mathbf{z}$中最大的k个元素的绝对值和“远大于”其余元素的绝对值和
- 如果$\mathbf{z}$是近似稀疏的，那么设恢复出来的是$\mathbf{z}'$，则$\mathbf{z}'$只是近似等于$\mathbf{z}$

有时这个$\mathbf{z}'$还“更好”，因为大体是去噪声后的$\mathbf{z}$

  - $\mathbf{z}'$与$\mathbf{z}$间的误差随着$\mathbf{z}$远离k-稀疏而越来越大

注意返回的$\mathbf{z}'$永远是k-稀疏的

  - 极端情况：$\mathbf{z}$完全不稀疏，则返回的$\mathbf{z}'$甚至可以与$\mathbf{z}$ “毫无关系”

需要的观测次数可以改进吗?

- 需要的线性观测次数$m = O(k\log(n/k))$是基本不可改进/最优的
- 这个量是哪来的? 一些直觉/类比:
  - 考虑基于比较的排序：n个元素有$n!$种排列，排序就是确定是哪一种排列
  - 每次比较可以排除一半可能性，总共就是$\log(n!) = O(n\log n)$次

类比：共有$\binom{n}{k}$个不同非0位置，每个线性观测去掉一半，需要

$$\log\binom{n}{k} = O(k\log(n/k))$$

实际情况更复杂，因为观测值是实数，
未必可以直接看作大于/小于这种2元情况

必须用高斯矩阵吗?

- 首先：未必使用高斯矩阵，后续科研探究了很多种其他类型的矩阵
  - 动机：在硬件实现中，高斯的型线性观测未必总易于实现
- 设计$\mathbf{A}$是nontrivial的；例如一种很自然、简单，但一定不行的矩阵：
  - 设$\mathbf{A}$是从$\mathbf{I}_n$选取若干随机行保留下来，其余元素都设置成0
  - $\mathbf{Az}$就是$\mathbf{z}$中只保留$\mathbf{A}$从$\mathbf{I}_n$选取的随机坐标
  - 但由于$\mathbf{z}$是稀疏的，大概率随机坐标都等于0，什么信息都没有留下！

压缩感知定理的推广：对于足够稠密、且“非病态”的$\mathbf{A}$，都可以用来恢复任意$\mathbf{z}$

给定“好”的$\mathbf{A}$，如何恢复$\mathbf{z}$?

转化成优化问题

- 压缩感知定理只是告诉我们如何选取$\mathbf{A}$，但之后如何恢复$\mathbf{z}$呢?
  - 回忆：$m < n$时$\mathbf{Ax} = \mathbf{b}$没有唯一解
- 思路：找非零坐标最少的$\mathbf{x}$

直接看来未必正确：这种$\mathbf{x}$可能也不等于$\mathbf{z}$，原则上可能找到更稀疏的；但精确版本的压缩感知定理确实证明这样选取就能找到$\mathbf{z}$

- 优化问题：

$$\begin{aligned} \min \quad & |\mathrm{supp}(\mathbf{x})| \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \end{aligned}$$

$\mathrm{supp}(\mathbf{x})$表示$\mathbf{x}$中非零坐标的集合
$|\mathrm{supp}(\mathbf{x})|$就是非零坐标的个数

关于 $|\mathrm{supp}(\mathbf{x})|$

类比ℓ_2：$\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$

$|\mathrm{supp}(\mathbf{x})|$ 又叫做$\mathbf{x}$向量的ℓ_0-范数

即非0的坐标个数，可以理解成 $\sum_i \mathbb{I}(x_i \neq 0)$，即非零都计贡献1

上页的优化问题也叫做ℓ_0-最小化问题

ℓ_0-最小化问题的求解?

- 但是ℓ_0-最小化问题是NP-hard的，也就是不能期望有多项式时间算法
- Idea：将ℓ_0“替换”成一种不NP-hard的，但又和ℓ_0足够接近的目标
- 我们选取：ℓ_1
  - 回忆ℓ_1的定义：$\|\mathbf{x}\|_1 = \sum_i |x_i|$

即每个坐标的绝对值求和
注意这个绝对值！

压缩感知定理：精确版

定理：设$\mathbf{A} \in \mathbb{R}^{m\times n}$其中$m = O(k\log(n/k))$且$\mathbf{A}$的每个元素都从$\mathcal{N}(0,1)$独立采样。则$\mathbf{A}$以高概率，同时对所有的$n$维的$k$-稀疏的$\mathbf{z}$，下面问题的唯一最优解恰好等于$\mathbf{z}$：

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_1 \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

为什么选ℓ_1?

- ℓ_1同时满足下面两个要求

  1. 最小化$\mathbf{x}$的ℓ_1-范数大致对应于寻找稀疏/1最少的$\mathbf{x}$
  2. ℓ_1最小化问题可以高效求解

- 作为对比：ℓ_0满足1但不满足2，ℓ_2满足2但不满足1

对比ℓ_1：ℓ_p-最小化问题的几何行为

为更好地理解ℓ_1的好处，这里对于几个典型p对比ℓ_p-最小化问题最优解的行为

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_p \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

ℓ_p-最小化

典型的p：$p = 1,2,\infty$

回忆ℓ_∞：$\|x\|_\infty = \max_i |x_i|$

ℓ_0不作讨论：ℓ_0虽然称作范数但并不是真正的范数

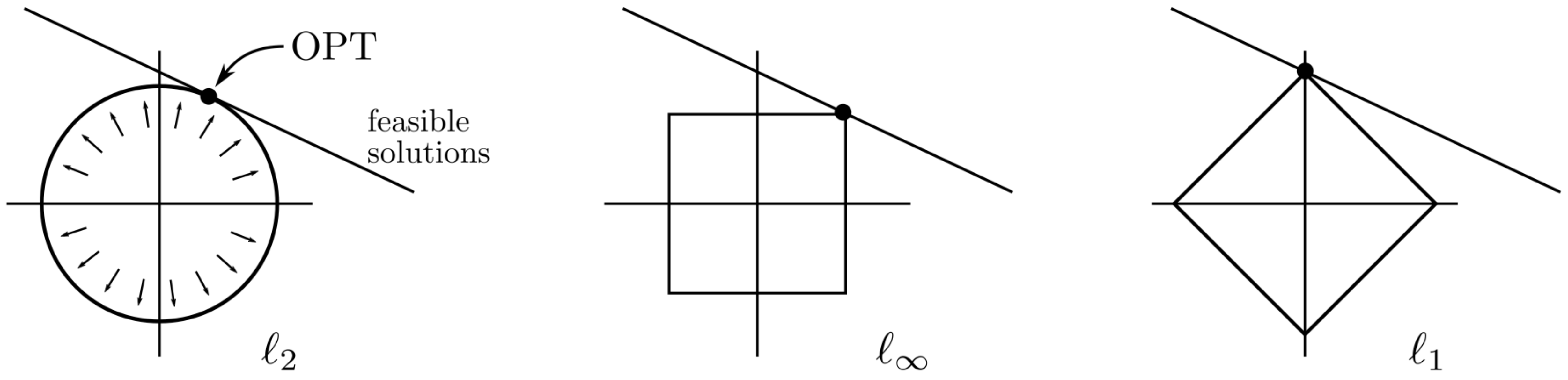
- 没有齐次性：$\|a\mathbf{x}\|_0 \neq |a| \|\mathbf{x}\|_0$

齐次性需要这里对任何a都取等号

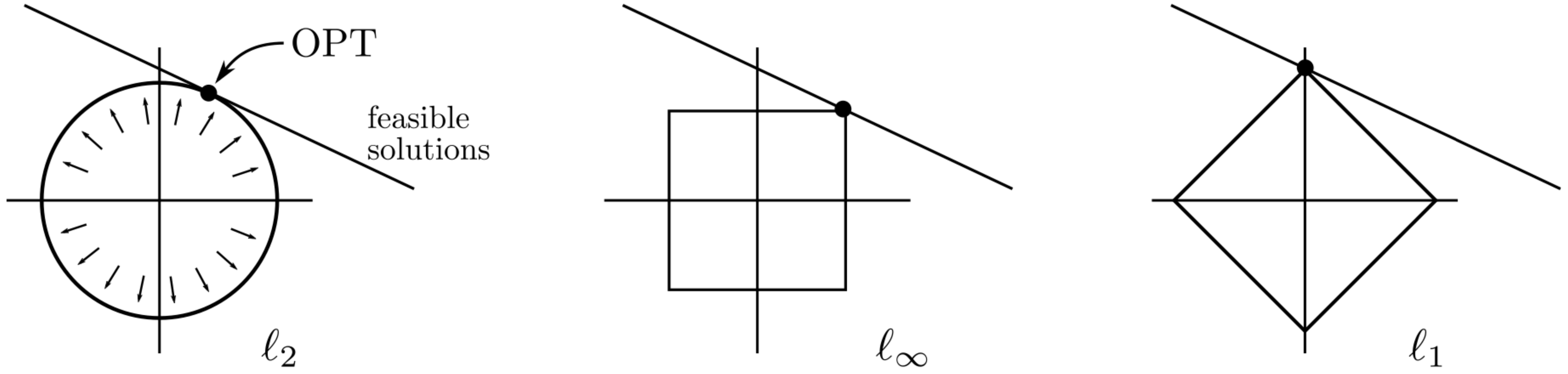
考虑二维的例子

$$
\begin{aligned}
\min \quad & \|\mathbf{x}\|_p \\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}
\end{aligned}
$$

- $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$是所有可行解的区域
- 考虑下面的例子：2维，且可行解区域就是一条直线，那么$\min \|x\|_p$如何取到?
- 从原点出发“吹ℓ_p气球“，问气球第一次碰到线的ℓ_p半径是多少



二维例子的最优解：ℓ_p球增长最快的方向



- ℓ_2：每个方向均匀增长，并不会考虑稀疏性；最优解通常异常稠密
- ℓ_∞：增长最快的是东北角，对应于全1向量，不稀疏
- ℓ_1：增长最快的是坐标轴方向，只有一个非零，很稀疏

思考：3维呢?

ℓ_1-最小化一般对应于稀疏解

* ℓ_p之间的关系

ℓ_∞可以表示一切n点的距离，但维度 = n

- 考虑n个数据点$(x_1, \ldots, x_n)$

叫做Fréchet's Embedding

- 定义$f(x_i) = (f_1(x_i), \ldots, f_n(x_i))$，其中$f_k(x_i) = d(x_k, x_i)$
- 验证：

$$\|f(x_i) - f(x_j)\|_\infty = \max_k |f_k(x_i) - f_k(x_j)| = \max_k |d(x_k, x_i) - d(x_k, x_j)|$$
$$= d(x_i, x_j)$$

最后的等号是为什么？

ℓ_1“包含”ℓ_2

- d维的ℓ_2可以用$O(d)$维的ℓ_1保$(1 \pm \epsilon)$倍距离表示

类似于JL

2.5.2 Lemma (Random projection from $\boldsymbol{\ell_2}$ to $\boldsymbol{\ell_1}$). *Let n, k be natural numbers, let $\varepsilon \in (0,1)$, and let us define a random linear map $T\colon \mathbb{R}^n \to \mathbb{R}^k$ by*

$$T(\mathbf{x})_i = \frac{1}{\beta k} \sum_{j=1}^{n} Z_{ij} x_j, \qquad i = 1, 2, \ldots, k,$$

where the Z_{ij} are independent standard normal random variables, and $\beta > 0$ is a certain constant ($\sqrt{2/\pi}$ if you must know). Then for every vector $\mathbf{x} \in \mathbb{R}^n$ we have

$$\mathrm{Prob}\Big[(1-\varepsilon)\|\mathbf{x}\|_2 \le \|T(\mathbf{x})\|_1 \le (1+\varepsilon)\|\mathbf{x}\|_2\Big] \ge 1 - 2e^{-c\varepsilon^2 k},$$

where $c > 0$ is a constant.

求解ℓ_1-最小化问题：线性规划

线性规划

Linear programming (LP)

关于术语：一般指代线性规划方法用linear programming，但具体指一个优化问题用linear program，与dynamic program/programming类似

- 变量：$x_1, \ldots, x_n \in \mathbb{R}$，可以自由变化，LP可以自动找到一组最优的$x$
- 约束：必须是线性约束，形如$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$

大于等于可以加负号实现；
等于可以小于等于 + 大于等于实现

- 目标函数：必须是线性的，形如$\min \sum_{j=1}^{n} c_j x_j$

max可以加负号实现

- 非线性的，例如x_j^2，$x_j x_k$，$\log(1 + x_j)$这些都不允许

线性规划的意义

- 线性规划是可以高效求解的
  - 理论上：存在最坏情况多项式时间算法
  - 实用上：形成了非常高效的solver

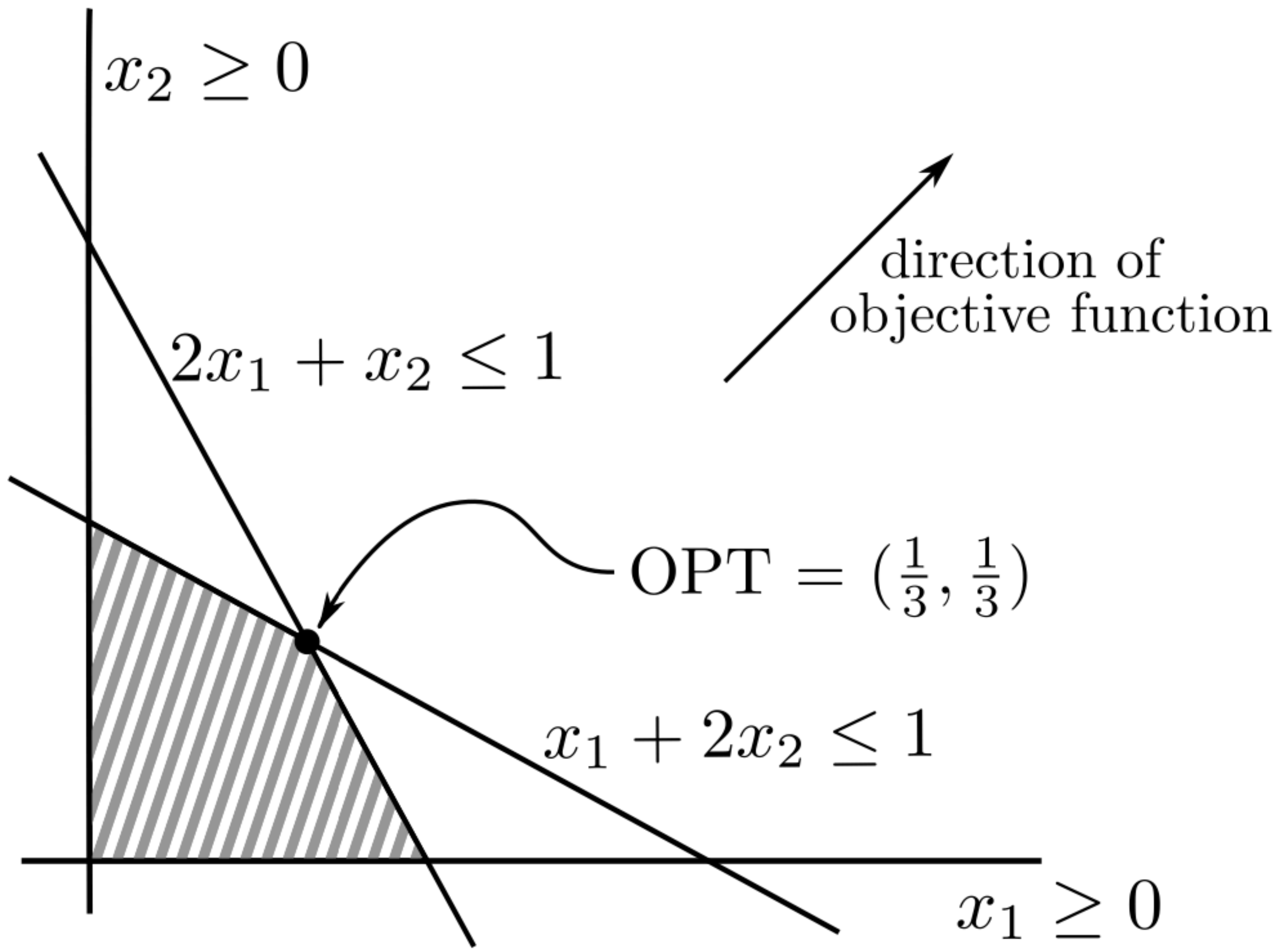
很多国家政府、企业、军队等都认识到LP的重要意义，几十年来的solver研发使得求解十分高效

- 线性规划可以用来建模/近似求解非常广泛的问题

由于其达到了很好的通用性和可计算性，是最重要的工具之一

一个简单例子与求解

$$
\begin{aligned}
\max \quad & x_1 + x_2 \\
\text{s.t.} \quad & x_1 \geq 0 \\
& x_2 \geq 0 \\
& 2x_1 + x_2 \leq 1 \\
& x_1 + 2x_2 \leq 1
\end{aligned}
$$



总体intuition：目标函数定义了优化的“方向”

目标就是在可行域里面找“优化方向”上的最远点

如何求解线性规划：理论上

- 理论上：存在多项式时间算法，例如ellipsoid method
- 假定存在一个单位时间可以调用的separation oracle，要求
  - 输入一组解，如果该解可行，那么输出Yes
  - 否则输出No，并给出一条不满足的约束
- ellipsoid method调用多项式次separation oracle以及进行多项式次其他操作

* 用Separation Oracle可以解一些“奇怪”的LP

- Ellipsoid配合separation oracle可能可以求解某些巨大的LP
- 巨大：变量有poly(n)个，然而约束有exp(n)个

$$
\begin{aligned}
\min \quad & \sum_{e\in E} c_e x_e \\
\text{s.t.} \quad & \sum_{e\in\delta(S)} x_e \geq 1, \forall S \subset V \\
& x_e \in [0,1], \quad \forall e \in E
\end{aligned}
$$

给定图$G=(V,E)$，$e\in E$边权c_e，该LP代表的是最小生成树

$\delta(S)$就是S个割边集

这约束的意思是，只要S不等于全集V，那么就必须选择一些跨越S的边，来保证连通性

虽然对MST本身没什么意义，但这类LP及其求解在带连通性约束的网络设计问题的近似算法研究中具有核心地位

问题：如何设计separation oracle？即如何判定给定的$\mathbf{x}$是可行解？

如何求解线性规划：实用上

- 实用上，ellipsoid method并不是最快的
- 相反，一种古老的称作simplex method的算法实际中表现很好
  - 但simplex method的最坏时间复杂度非常高，可以是$\exp(n)$的！
  - 因此这侧面说明实际上一般遇不到最坏情况

对Simplex Method实际情况表现好的理论解释

奖给最杰出的理论计算机科学论文（的作者）

- [Spielman-Teng, JACM 04]，Gödel Prize 哥德尔奖

提出了smoothed analysis框架，并且应用在了单纯形法等重要常用算法

- 我们一般考虑算法时间复杂度都是考虑最坏情况
- Smoothed analysis中，假设输入并非最坏
  - 而是任意一个输入再加上一个随机噪声
- ST04证明的是在期望意义下，单纯形法是多项式时间复杂度

随机性来自于噪声

完美契合了实际情况：实际上LP的输入都有噪声

实际该使用的：成熟的Solver软件/类库

自己实现的很难比得上solver的，因为solver的实现是无数特例研究/工程优化等的结果

- 刚刚的讨论主要停留在理论/学术上
- 真正要解LP，还是要尽量使用已经实现好的成熟solver
    - 商业求解器：CPLEX，Gurobi
    - 开源求解器：GLPK (GNU Linear Programming Kit)，lp_solve
    - 一些软件/类库带的求解器：Matlab, numpy/scipy

lp_solve

- 性能中等，开源，较易使用的LP solver（也支持整数规划！）
- https://lpsolve.sourceforge.net/
- 安装后可以使用命令行工具lp_solve LP_FILE.lp
- LP_FILE的格式、样例： https://lpsolve.sourceforge.net/5.5/lp-format.htm

```
/* xxx.lp */

max: 143 x + 60 x1;

120 x + 210 x1 <= 15000;
110 x + 30 x1 <= 4000;
x + x1 <= 75;
```

xxx.lp文件

```
Value of objective function: 6315.62500000

Actual values of the variables:
x                          21.875
x1                         53.125
```

运行lp_solve xxx.lp

注释

```
/* xxx.lp */
```

变量命名规则类似C/C++，支持浮点数系数

优化objective，可选max或min

```
max: 143 x + 60 x1;
```

这些是约束，可以用<, <=, >, >=, =

```
120 x + 210 x1 <= 15000;
110 x + 30 x1 <= 4000;
x + x1 <= 75;
```

不要忘记给变量设置范围：默认是[0, +∞)；如果要(-∞, +∞)可以写free x1

回到主题：如何用LP求解ℓ_1-最小化问题?

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_1 \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

- 如何转化成LP?

因为有绝对值!

- $\mathbf{A}\mathbf{x} = \mathbf{b}$已经是线性约束了，但$\|x\|_1 = \sum_i |x_i|$并不是线性的

- 先考虑一个简单情况：若真实信号$\mathbf{z}$每维 ≥ 0，则可仅考虑$\mathbf{x} \geq 0$的情况，得到

$$\begin{aligned} \min \quad & \|\mathbf{x}\|_1 = \sum_i x_i \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

有$x_i \geq 0$的假设，此处可去掉绝对值

一般情况：如何处理绝对值

添加辅助变量**y**来刻画“绝对值”，改写成

$$
\begin{aligned}
\min \quad & \sum_i y_i \\
\text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\
& y_i - x_i \geq 0, \quad \forall i \\
& y_i + x_i \geq 0, \quad \forall i
\end{aligned}
$$

否则若$y_i > |x_i|$则可以降低y_i到$y_i = |x_i|$，不影响其他任何变量，但降低了目标值

在这个新LP中：对每个i，有$y_i \geq |x_i|$，并且由于目标是min，必然有$y_i = |x_i|$

小扩展：输入有噪声

- 线性规划一般可以轻易处理有噪声的情况
- 例如，我们收到的不是精确的$\mathbf{b} = \mathbf{A}\mathbf{z}$，而是收到一个$\hat{\mathbf{b}}$
  - 每个维度上$\hat{b}_i \in [b_i - \epsilon, b_i + \epsilon]$
- 如果把$\hat{\mathbf{b}}$当$\mathbf{b}$来用，甚至可能导致LP无解

对每个$1 \leq i \leq m$都需要有

解决方案：将$\mathbf{A}\mathbf{x} = \mathbf{b}$换成$\sum_{j=1}^{n} a_{ij}x_j \leq b_i + \epsilon$和$\sum_{j=1}^{n} a_{ij}x_j \geq b_i - \epsilon$两组

压缩感知定理也有对应的带噪声的版本

基于压缩感知的Sparse Recovery：完整算法

- 预先给定的参数：k，n，代表信号是n维的k-稀疏的实向量
- 算法先生成一个$m \times n$的随机高斯矩阵$\mathbf{A}$，其中$m = O(k\log(n/k))$
- 算法得到$\mathbf{b} \in \mathbb{R}^m$，使得$\mathbf{b} = \mathbf{A}\mathbf{z}$
- 算法继续计算右边LP的最优解$\mathbf{x}$作为输出

每一维都是独立标准正态$\mathcal{N}(0,1)$

压缩感知定理告诉我们大概率有$\mathbf{x} = \mathbf{z}$

$$
\begin{aligned}
\min \quad & \sum_i y_i \\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\
& y_i - x_i \geq 0, \quad \forall i \\
& y_i + x_i \geq 0, \quad \forall i
\end{aligned}
$$

作业十八：压缩感知实验报告

- 本次作业为实验报告，具体要求请见下面链接中的PDF文档：

https://disk.pku.edu.cn/link/AA9B3CB98E746C4490B83C117BA1426408

作业需要调用一个解LP的算法

本次实验报告不限制编程语言和LP solver

  - 可以使用lp_solve（或其他solver），也可以自己实现一个解LP的算法

- 截止日期：2024年6月12日，请提交到教学网！

单纯形法简介：举例讲解

参考： https://math.mit.edu/~goemans/18310S15/lpnotes310.pdf

预处理：转化成“标准型”

- 标准型的要求：目标是max，约束只有等式，所有变量非负
- 转换方法：
  - 如果目标是$\min z$，则转成$\max -z$
  - 如果约束是$a_{i1}x_1 + \ldots + a_{in}x_n \le b_i$，引入非负松弛变量$s_i$，改成两个约束：$s_i \ge 0$，以及$a_{i1}x_1 + \ldots + a_{in}x_n + s_i = b_i$

    如果是≥应该如何操作?
  - 如果有变量x_i没有符号约束，将它替换成$x_i' - x_i''$，$x_i' \ge 0$，$x_i'' \ge 0$

转化成标准型举例

$$\begin{aligned} \min \quad & -2x_1 + 3x_2 \\ & x_1 - 3x_2 + 2x_3 \le 3 \\ & -x_1 + 2x_2 \ge 2 \\ & x_1 \in \mathbb{R}, x_2, x_3 \ge 0 \end{aligned}$$

变max，改等号

$$\begin{aligned} \max \quad & 2x_1 - 3x_2 \\ & x_1 - 3x_2 + 2x_3 + s_1 = 3 \\ & -x_1 + 2x_2 - s_2 = 2 \\ & x_1 \in \mathbb{R}, x_2, x_3, s_1, s_2 \ge 0 \end{aligned}$$

替换无符号限制变量x_1

$$\begin{aligned} \max \quad & 2(x_1' - x_1'') - 3x_2 \\ & (x_1' - x_1'') - 3x_2 + 2x_3 + s_1 = 3 \\ & -(x_1' - x_1'') + 2x_2 - s_2 = 2 \\ & x_1', x_1'', x_2, x_3, s_1, s_2 \ge 0 \end{aligned}$$

单纯形法：举例讲解

考虑已化为标准型LP：

$$
\begin{aligned}
\max \quad & x_1 + x_2 \\
& 2x_1 + x_2 + x_3 = 4 \\
& x_1 + 2x_2 + x_4 = 3 \\
& x_1, x_2, x_3, x_4 \geq 0
\end{aligned}
$$

得到系数矩阵/方程组

将所有等号约束以及z方程（目标函数）放在一起得到一个方程组

$$
\begin{array}{ccccccccc}
z & -x_1 & -x_2 & & & = & 0 & & \text{Row 0} \\
 & 2x_1 & +x_2 & +x_3 & & = & 4 & & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & & \text{Row 2}
\end{array}
$$

时刻注意：所有变量$x_1, x_2, x_3, x_4 \geq 0$，要寻找的是最大的$z$

额外要求

$$
\begin{array}{llllllll}
z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\
 & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2}
\end{array}
$$

m是约束/行数

该$\mathbf{I}_m$是$m \times m$单位阵，从忽略Row 0的矩阵找

额外要求：等号右边都 ≥ 0，系数矩阵含$\mathbf{I}_m$且该$\mathbf{I}_m$对应的列的系数在Row 0都是0

上面的例子满足这个要求：$\mathbf{I}_m$是x_3, x_4对应的

一般如何达到该要求后面会讲解

单纯形法框架

- 从刚刚得到的满足额外要求的标准型开始，单纯形法进行若干步迭代
- 我们主要描述每次迭代需要做的算法步骤，包括：
  - 如何判定终止——终止条件即最优条件
  - 当前未终止的话，这步内部做什么

当然还有判定“无解”“有无穷多解”等各种细节，在此不讨论

定义“基本可行解”

$$\begin{array}{cccccccl} z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\ & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\ & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2} \end{array}$$

基变量：某列上唯一系数非0的变量，这里是x_3, x_4

非基变量：其他变量，这里是x_1, x_2

基本可行解：将非基变量设置成0，代入后解方程得出基变量值

上面例子：非基$x_1 = x_2 = 0$，基$x_3 = 4$，$x_4 = 3$

将目标用非基变量表示

$$
\begin{array}{lllllll}
z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\
 & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2}
\end{array}
$$

- 在这个例子中，x_1, x_2是非基变量，Row 0已经是只有非基变量，不需要变换
  - 否则，总可以将每个基变量用非基变量线性表示，然后替换
  - 替换后Row 0只含有非基变量

重要规则1：最优解/停止判据

$$
\begin{array}{cccccccl}
z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\
 & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2}
\end{array}
$$

- 将目标Row 0用非基变量表示后，如果某变量系数是负的
    - 则一定可以增大变量值来改进z的值，也就是此时非最优
- 因此最优/停止判据：Row 0所有系数都是正的

选择某个非基作为entering变量

$$
\begin{array}{lllllcll}
z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\
 & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2}
\end{array}
$$

- 现在根据规则1不是最优解

选取下标最小的（最靠左的），又叫Bland规则，用于防止一些情况下单纯形法死循环

- 下一步：选某Row 0行系数为负的非基变量作为entering（加入）变量
- 选某个行作为pivot行高斯消元

如何选行，也就是pivot，也很重要，规则后面讲

  - 消元后就会有某个基变量成为非基变量，称为leaving变量

leaving变量具体是谁不重要，也不需要我们去选择，只是可以证明有一个enter的非基变量就必有一个leaving的基变量

先假设pivot选好了：进行高斯行变换

- 选x_1作为加入变量，选Row 1作为pivot运行消元

$$
\begin{array}{lllllll}
z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\
 & \mathbf{2x_1} & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2}
\end{array}
$$

$$
\begin{array}{lllllll}
z & & -\frac{1}{2}x_2 & \frac{1}{2}x_3 & & = & 2 & \text{Row 0} \\
 & x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}x_3 & & = & 2 & \text{Row 1} \\
 & & \frac{3}{2}x_2 & -\frac{1}{2}x_3 & +x_4 & = & 1 & \text{Row 2}
\end{array}
$$

新的基本可行解

得到新的基本可行解就完成了一轮迭代
此时可以利用最优化/停止判据决定是否继续
若未判定停止：则继续利用刚才的规则

$$
\begin{array}{lllllll}
z & & -\frac{1}{2}x_2 & \frac{1}{2}x_3 & & = & 2 & \text{Row 0} \\
& x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}x_3 & & = & 2 & \text{Row 1} \\
& & \frac{3}{2}x_2 & -\frac{1}{2}x_3 & +x_4 & = & 1 & \text{Row 2}
\end{array}
$$

- 新的基变量是x_1, x_4，令非基变量为0
- 得到$x_2 = x_3 = 0$，$x_1 = 2$，$x_4 = 1$，$z = 2$

依然满足：等号右边都 ≥ 0，系数矩阵含$\mathbf{I}_m$且该$\mathbf{I}_m$对应的列的系数在Row 0都是0

重要规则2：如何为entering变量选行/pivot

——补全缺失的最后一环

- 除去Row 0的行中找那些让entering变量系数严格 > 0的，将RHS除以这个系数
- 选比值比值最小的行

$$\begin{array}{rrrrrcll} z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\ & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\ & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2} \end{array}$$

- 例如，选x_1作为entering，Row 1比值2，Row 2比值3，应该选Row 1

为什么要选比值最小的行？选其他行会怎样?

- 比如选Row 2会怎样?

$$
\begin{array}{lllllll}
z & -x_1 & -x_2 & & & = & 0 & \text{Row 0} \\
 & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row 1} \\
 & \mathbf{x_1} & +2x_2 & & +x_4 & = & 3 & \text{Row 2}
\end{array}
$$

$$
\begin{array}{lllllll}
z & & +x_2 & & +x_4 & = & 3 & \text{Row 0} \\
 & & -3x_2 & +x_3 & -2x_4 & = & -2 & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row 2}
\end{array}
$$

看上去好像最优了?

选其他行的问题所在

单纯形法重要要求：每次的基本解必须得到可行解

$$
\begin{array}{ccccccccl}
z & & +x_2 & & +x_4 & = & 3 & & \text{Row 0} \\
 & & -3x_2 & +x_3 & -2x_4 & = & -2 & & \text{Row 1} \\
 & x_1 & +2x_2 & & +x_4 & = & 3 & & \text{Row 2}
\end{array}
$$

- 此时基本解$x_2 = x_4 = 0$，$x_1 = 3$，$x_3 = -2$，非可行解！

本质原因：等号右边有负数
导致负数的原因是选了Row 2，比值大于Row 1

思考：为什么选了比值最小的行就能确保RHS
不会有负数了？

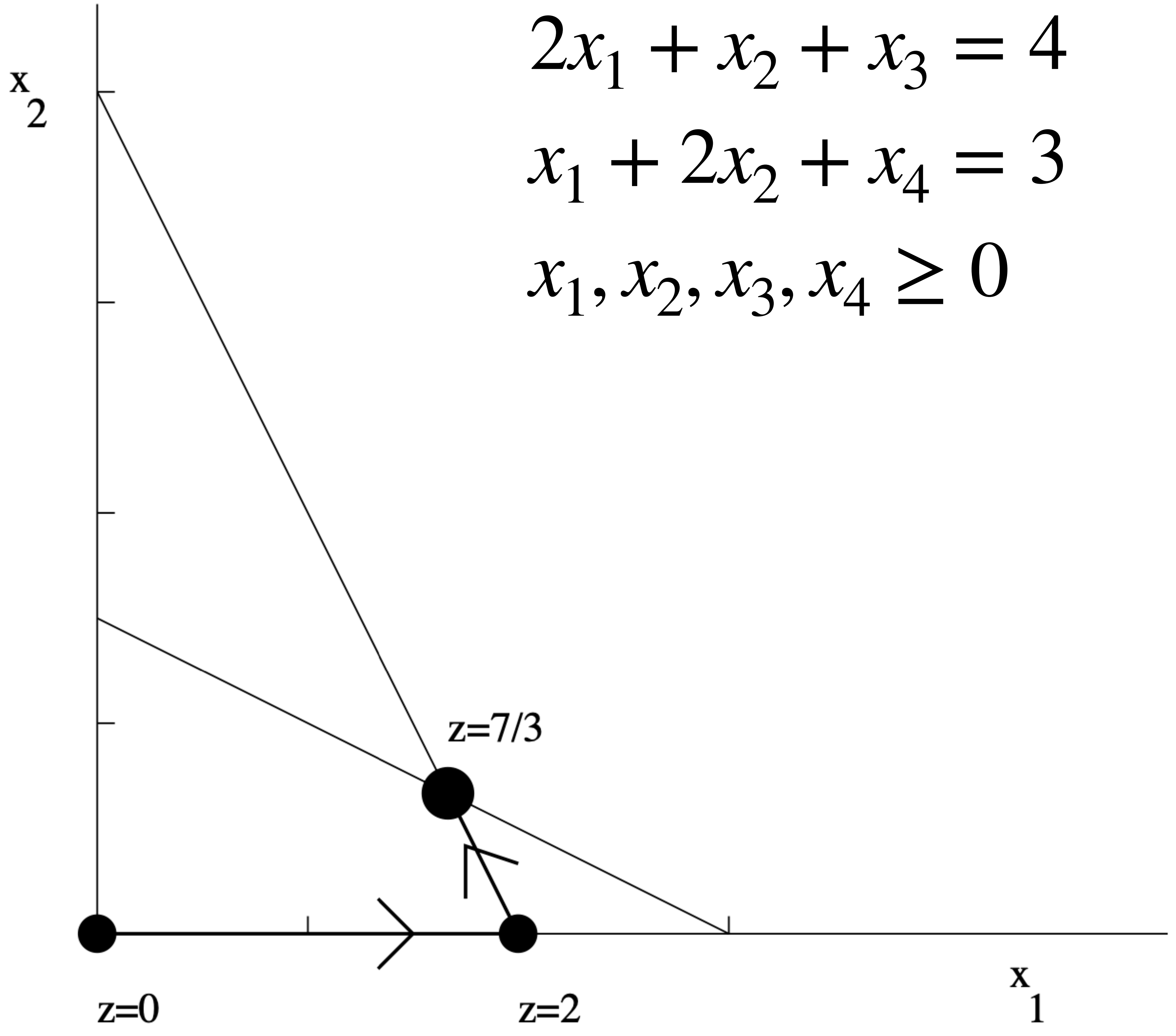
执行过程的几何直观

$$\begin{array}{llllcl} z & -x_1 & -x_2 & & = & 0 \quad \text{Row 0} \\ & \mathbf{2x_1} & +x_2 & +x_3 & = & 4 \quad \text{Row 1} \\ & x_1 & +2x_2 & \quad +x_4 & = & 3 \quad \text{Row 2} \end{array}$$

$$\begin{array}{llllcl} z & & -\frac{1}{2}x_2 & \frac{1}{2}x_3 & = & 2 \quad \text{Row 0} \\ & x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}x_3 & = & 2 \quad \text{Row 1} \\ & & \mathbf{\frac{3}{2}x_2} & -\frac{1}{2}x_3 \quad +x_4 & = & 1 \quad \text{Row 2} \end{array}$$

$$\begin{array}{llllcl} z & & +\frac{1}{3}x_3 & +\frac{1}{3}x_4 & = & \frac{7}{3} \quad \text{Row 0} \\ & x_1 & +\frac{2}{3}x_3 & -\frac{1}{3}x_4 & = & \frac{5}{3} \quad \text{Row 1} \\ & x_2 & -\frac{1}{3}x_3 & +\frac{2}{3}x_4 & = & \frac{2}{3} \quad \text{Row 2} \end{array}$$

$$\begin{array}{ll} \max & x_1 + x_2 \\ & 2x_1 + x_2 + x_3 = 4 \\ & x_1 + 2x_2 + x_4 = 3 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$



“去掉”额外要求

额外要求：等号右边都 ≥ 0，系数矩阵含$\mathbf{I}_m$且该$\mathbf{I}_m$对应的列的系数在Row 0都是0

先定义一个辅助的所谓“Phase I问题”

对每个非Row 0的Row i：若$b_i < 0$则两边先取负

- 引入一个新变量t_i，系数是1，加在这一行上
- 将目标函数改为$\min \sum_i t_i$，并替换掉以前的Row 0

这个新问题中，最优解一定是所有$t_i = 0$，最优值 = 0

举例

$$\begin{aligned} \max \quad & 2x_1 + 3x_2 + 4x_3 \\ & 3x_1 + 2x_2 + x_3 = 10 \\ & 2x_1 + 5x_2 + 3x_3 = 15 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

写成方程组

$$\begin{array}{cccccccl} z & 2x_1 & +3x_2 & +4x_3 & = & 0 & & \text{Row 0} \\ & 3x_1 & +2x_2 & +x_3 & = & 10 & & \text{Row 1} \\ & 2x_1 & +5x_2 & +3x_3 & = & 15 & & \text{Row 2} \end{array}$$

加入辅助变量、改成新的目标

$$\begin{array}{cccccccccl} z & & & & -t_1 & -t_2 & = & 0 & & \text{Row 0} \\ & 3x_1 & +2x_2 & +x_3 & +t_1 & & = & 10 & & \text{Row 1} \\ & 2x_1 & +5x_2 & +3x_3 & & +t_2 & = & 15 & & \text{Row 2} \end{array}$$

还差一小步

- 在这个新问题中，还需要将每个Row i $(i \geq 1)$加到Row 0上才满足额外要求

$$
\begin{array}{cccccccccl}
z & & & & -t_1 & -t_2 & = & 0 & & \text{Row 0} \\
 & 3x_1 & +2x_2 & +x_3 & +t_1 & & = & 10 & & \text{Row 1} \\
 & 2x_1 & +5x_2 & +3x_3 & & +t_2 & = & 15 & & \text{Row 2}
\end{array}
$$

将Row 1和Row 2加到Row 0

此时得到的问题满足单纯形法所有要求！

$$
\begin{array}{cccccccccl}
z & 5x_1 & 7x_2 & 4x_3 & & & = & 25 & & \text{Row 0} \\
 & 3x_1 & +2x_2 & +x_3 & +t_1 & & = & 10 & & \text{Row 1} \\
 & 2x_1 & +5x_2 & +3x_3 & & +t_2 & = & 15 & & \text{Row 2}
\end{array}
$$

Phase I问题和Phase II问题

回忆：此时的最优解一定是让辅助变量t_i都等于0的解

- 在刚才的新的Phase I问题上用单纯形法求解，直到得到最优解
  - 若所有辅助变量都是非基，则可以都删除，剩下的原变量满足额外要求
- 否则：存在一个辅助基变量t_i，此时基本解中$t_i = 0$，这推出$b_i = 0$
  - 随便挑选一个第i行系数非0的原变量x_j为pivot进行变换

这里假定唯一系数非零的t_i出现在第i行，b_i是对应的RHS

反复运行后，就可以让所有辅助变量称为非基变量

因为$b_i = 0$，不会对解产生任何影响，但却可以将t_i变成非基

- 之后可以在原问题上照常运行单纯形法（称为Phase II）

额外要求：等号右边都 ≥ 0，系数矩阵含$\mathbf{I}_m$且该$\mathbf{I}_m$对应的列的系数在Row 0都是0

Min Vertex Cover

- 输入：无向无权图$G = (V, E)$
- 目标：找一个最小的子点集$S \subseteq V$使得任何一条边$uv \in E$有至少一个顶点在S
  - 即$u \in S$或$v \in S$，可以看作是S“覆盖”了所有边
- NP-hard，目前最好的是2-近似，即找到一个S大小至多是最优的2倍

假定Unique Game Conjecture (UGC)，2-近似就是最优的

写成整数线性规划

$$\min \sum_i x_i$$

$$x_i + x_j \geq 1 \quad \forall (i,j) \in E$$

$$x_i \in \{0,1\} \quad \forall i$$

表示每条边都被覆盖

x只能取0或者1，因此是整数规划

x_i称为决策变量，因为决策每个点要不要放在S里

- 整数规划的解与S一一对应：$S := \{i : x_i = 1\}$，并且目标函数是$|S|$
  - 因此该整数线性规划等价于原Min Vertex Cover问题

也因此依然是NP-hard：只是用另一种形式转述了问题，没简化问题

关键步骤：线性规划松弛

称作ILP，I = Integer

$$\min \sum_i x_i$$
$$x_i + x_j \geq 1 \quad \forall (i,j) \in E$$
$$x_i \in \boxed{\{0,1\}} \quad \forall i$$

称作LP

$$\min \sum_i x_i$$
$$x_i + x_j \geq 1 \quad \forall (i,j) \in E$$
$$x_i \in \boxed{[0,1]} \quad \forall i$$

- 决策变量x_i从$\{0,1\}$整数值放宽/松弛成$[0,1]$实数
  - 变成LP，可高效求解了！

    LP不再NP-hard
  - 但是求出来的解却不是$\{0,1\}$了，如何对应回原来的解呢？

ILP与LP松弛的关系

$$
\begin{aligned}
\min \quad & \sum_i x_i \\
& x_i + x_j \geq 1 \quad \forall (i,j) \in E \\
& x_i \in \{0,1\} \qquad \forall i
\end{aligned}
$$

$$
\begin{aligned}
\min \quad & \sum_i x_i \\
& x_i + x_j \geq 1 \quad \forall (i,j) \in E \\
& x_i \in [0,1] \qquad \forall i
\end{aligned}
$$

- 观察1：任何ILP的可行解都是LP的可行解
- 观察2：任何解上的目标函数值都相等，等于 $\sum_i x_i$
- 因此：设LP的最优是LP*，ILP的最优是ILP*，则$\text{LP*} \leq \text{ILP*}$

将LP的最优解转化回“整数”解：Rounding

$$\min \quad \sum_i x_i$$
$$x_i + x_j \geq 1 \quad \forall (i,j) \in E$$
$$x_i \in [0,1] \quad \forall i$$

- 算法：解LP，设最优解是$\mathbf{x}^*$，定义$S := \{x_i^* : x_i^* \geq 0.5\}$
- 可以证明：S就是2-近似的！

可以理解成round-up：
四舍五入

证明S是2-近似的

$$\min \quad \sum_i x_i$$
$$x_i + x_j \geq 1 \quad \forall (i,j) \in E$$
$$x_i \in [0,1] \quad \forall i$$

算法：解LP，设最优解是$\mathbf{x}^*$，定义$S := \{x_i^* : x_i^* \geq 0.5\}$

首先，S是合法的vertex cover：

- 根据LP约束，任何$(i,j) \in E$有$x_i^* + x_j^* \geq 1$，必有至少一个≥ 0.5，选进了S

然后，注意到$|S| = \sum_i \mathbb{I}(x_i^* \geq 0.5) \leq \sum_i 2x_i^* = 2 \cdot \mathrm{LP}^* \leq 2 \cdot \mathrm{ILP}^*$

回忆：LP松弛与ILP的关系

证毕

Set Cover

- 输入：m个$\{1,\ldots,n\}$上的集合$S_1,\ldots,S_m$
- 目标：找到最小的$\{1,\ldots,n\}$的子集C使得$\forall 1 \le k \le m$有$C \cap S_k \neq \emptyset$
- 线性规划

每个$\{1,\ldots,n\}$的元素i对应一个决策变量x_i
求和对应选取的元素的个数

$$\min \quad \sum_{i=1}^{n} x_i$$

对每个集合S_k，都要求至少有一个元素被选中

$$\sum_{i \in S_k} x_i \ge 1 \quad \forall 1 \le k \le m$$

$$0 \le x_i \le 1$$

如果$x_i \in \{0,1\}$则为ILP并精确对应原问题，这里是LP松弛

由于是松弛，可得：$\mathrm{LP}^* \le \mathrm{OPT}$

随机Rounding算法

达到$O(\log m)$近似比

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} x_i \\
& \sum_{i \in S_k} x_i \ge 1 \quad \forall 1 \le k \le m \\
& 0 \le x_i \le 1
\end{aligned}
$$

- 设已经解得LP的最优解是$\mathbf{x}^*$，此时可把$0 \le x_i \le 1$理解成概率
- 算法单位步骤：对每个i，以x_i概率选取，并放进C

每个元素可能会重复放入C，此时仅保留一份

- 将算法单位步骤重复独立运行$O(\log m)$次
- 近似比分析：

$$\mathbb{E}[\,|C|\,] = O(\log m) \cdot \sum_{i=1}^{n} x_i \le O(\log m) \cdot \mathrm{OPT}$$

利用了LP$^* \le \mathrm{OPT}$

随机Rounding算法

达到$O(\log m)$近似比

$$\min \sum_{i=1}^{n} x_i$$
$$\sum_{i\in S_k} x_i \geq 1 \quad \forall 1 \leq k \leq m$$
$$0 \leq x_i \leq 1$$

- 设已经解得LP的最优解是$\mathbf{x}^*$，此时可把$0 \leq x_i \leq 1$理解成概率
- 算法单位步骤：对每个i，以x_i概率选取，并放进C
- 将算法单位步骤重复独立运行$O(\log m)$次

每个元素可能会重复放入C，此时仅保留一份

- 解的可行性分析：对集合S_k，所有$O(\log m)$次试验都没选任何S_k元素的概率：

$$(\prod_{i\in S_k}(1-x_i))^{O(\log m)} \leq \exp(-\log m \sum_{i\in S_k} x_i) \leq O(1/m)$$

- 用union bound，存在一个集合S_k没元素被选的概率至多$O(1/m)\cdot m = O(1)$