



第八章 随机算法

骆吉洲
计算机科学与技术学院



提要

- 8.1 *Introduction to Randomized Algorithms*
- 8.2 *Randomized Numerical Algorithms*
- 8.3 *Randomized Selection Algorithm*
- 8.4 *Randomized Algorithm for Prime Test*
- 8.5 *Randomized Sorting Algorithm*
- 8.6 *Randomized Min-Cut Algorithm*
- 8.7 附录：概率基础



8.1 Introduction to Randomized Algorithms

- 随机算法的基本概念
- 随机算法的分类
- 随机算法的性能分析方法



随机算法的基本概念

- 什么是随机算法
 - 随机算法是一种使用概率和统计方法在其执行过程中对于下一计算步骤作出随机选择的算法
- 随机算法的优越性
 - 对于有些问题：算法简单
 - 对于有些问题：时间复杂性低
 - 对于有些问题：同时具有简单和时间复杂性低
- 随机算法的随机性
 - 对于同一实例的多次执行，效果可能完全不同
 - 时间复杂性的一个随机变量
 - 解的正确性和准确性也是随机的



随机算法的分类

- 随机数值算法
 - 主要用于数值问题求解
 - 算法的输出往往是近似解
 - 近似解的精确度与算法执行时间成正比
- Monte Carlo算法
 - 主要用于求解需要准确解的问题
 - 算法可能给出错误解
 - 获得精确解概率与算法执行时间成正比



• Las Vegas算法

- 一旦找到一个解，该解一定是正确的
- 找到解的概率与算法执行时间成正比
- 增加对问题反复求解次数，可是求解无效的概率任意小

• Sherwood算法

- 一定能够求得一个正确解
- 确定算法的最坏与平均复杂性差别大时，加入随机性，即得到Sherwood算法
- 消除最坏行为与特定实例的联系



随机算法的性能分析



• 随机算法分析的特征

- 仅依赖于随机选择，不依赖于输入的分布
- 确定算法的平均复杂性分析：

 - 依赖于输入的分布
 - 对于每个输入都要考虑算法的概率统计性能

- 随机算法分析的目标

 - 平均时间复杂性：时间复杂性随随机变量的均值
 - 获得正确解的概率
 - 获得优化解的概率
 - 解的精确度估计

8.2 数值随机算法

- 计算 π 值
- 计算定积分



计算 π 值



• 数学基础

- 设有一个半径为 r 的圆及其外切四边形



- 向正方形随机地投掷 n 个点，设 k 个点落入圆内
- 投掷点落入圆内的概率为 $(\pi r^2)/(4r^2) = \pi/4$.
- 用 k/n 逼近 $\pi/4$ ，即 $k/n \approx \pi/4$ ，于是 $\pi \approx (4k)/n$.
- 我们可以令 $r=1$ 用投掷 n 个点的方法计算 π

• 算法

1. $K=0;$
2. For $i=1$ To n Do
3. 随机地产生四边形中的一点 $(x, y);$
4. If $x^2+y^2 \leq 1$ Then $k=k+1;$
5. Return $(4k)/n$

• 时间复杂性= $O(n)$

- 不是输入的大小，而是随机样本的大小
- 解的精确度

 - 随着随机样本大小 n 增加而增加

问题：样本数 n 和精度之间能建立关联关系吗？



计算定积分

• 问题

- 计算积分 $\int_a^b g(x)dx$

• 数学基础

- 令 $f(x)$ 是区间 $[a, b]$ 上的一组独立、同分布的随机变量 $\{\xi_i\}$ 的任意密度函数
- 令 $g^*(x)=g(x)/f(x)$ ，则 $\{g^*(\xi_i)\}$ 是密度为 $f(x)$ 的随机变量集合，而且

$$E(g^*(\xi_i)) = \int_a^b g^*(x)f(x)dx = \int_a^b g(x)dx = I$$

$$E(g^*(\xi_i)) = \int_a^b g^*(x)f(x)dx = \int_a^b g(x)dx = I$$

- 由强大数定律 $\Pr\left(\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n g^*(\xi_i) = I\right) = 1$

- 我们可以用 $\left(\frac{1}{n} \sum_{i=1}^n g^*(\xi_i)\right)$ 来近似计算 I

- 令 $f(x)=1/(b-a)$ $a \leq x \leq b$

- 索求积分可以由如下 I' 来近似计算 I

$$I' = \frac{1}{n} \sum_{i=1}^n g^*(\xi_i) = \frac{1}{n} \sum_{i=1}^n g(\xi_i)/f(\xi_i) = \frac{1}{n} \sum_{i=1}^n (b-a)g(\xi_i)$$



• 算法

1. $I=0;$
 2. For $i=1$ To n
 3. 随机产生 $[a, b]$ 中点 x ;
 4. $I=I+g(x);$
 5. Return $(b-a)*I/n$
- 时间复杂性= $O(n)$
 - 不是输入的大小, 而是随机样本的大小
 - 解的精确度
 - 随着随机样本大小 n 增加而增加

问题: 样本数 n 和精度之间能建立关联关系吗? 😊



8.3 随机选择算法

- 问题的定义
- 随机算法
- 算法的性能分析
- Las Vegas算法



问题的定义

- 输入: $S=\{x_1, x_2, \dots, x_n\}$, 整数 k , $1 \leq k \leq n$.
- 输出: S 中第 k 小元素.

记号

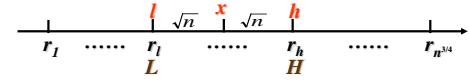
$\text{Rank}(Q, i) = \text{集合 } Q \text{ 中的元素 } i \text{ 的 rank}$
(第 k 小元素的rank是 k)
 $\min(Q, i) = \text{集合 } Q \text{ 中第 } i \text{ 个最小元素.}$



随机算法

• 基本思想

- 从 S 中随机地抽取 $n^{3/4}$ 个样本存入 R , 排序 R
- S 中第 k 最小元素可能成为 R 中 $x=kn^{3/4}/n$ 最小元素
- 为了解决误差问题, 我们考察区间 $[x-n^{1/2}, x+n^{1/2}]$



- 把 S 中属于 $[L, H]$ 数据存入 P
- 在 P 中查找 $\min(S, k)$

LAZYSELECT(S, k)

1. R =独立、均匀、可放回地从 S 随机选取的 $n^{3/4}$ 元素;
2. 在 $O(n)$ 时间内排序 R ;
3. $x=(k/n)n^{3/4}; /* (k/n)n^{3/4}=kn^{-1/4} */$
4. $l=\max\{\lfloor x-\sqrt{n} \rfloor, 0\}; h=\min\{\lfloor x+\sqrt{n} \rfloor, n^{3/4}\};$
5. $L=\min(R, l); H=\max(R, h);$
6. $L_p=\text{Rank}(S, L), H_p=\text{Rank}(S, H); /* L和H与S元素比较 */$
7. $P=\{y \in S \mid L \leq y \leq H\};$
8. If $\min(S, k) \in P$ and $|P| \leq 4n^{3/4}+1$
 - * max(S, k) ∈ P 可由 $L_p \leq k \leq H_p$ 确定 *
9. Then 排序 P , $\min(S, k)=\min(P, (k-L_p))$, 算法结束;
10. ELSE goto 1



算法的性能分析

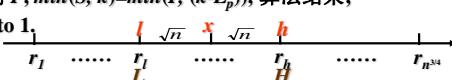
• 数学基础

– 数学期望

- 离散随机变量 X 的数学期望 $E[X]=\sum_i i \times P(X=i)$
- 若 $f(x)$ 是定义在整数集上的实数值函数, 则 $E[f(X)]=\sum_i f(i) \times P(X=i).$

– Markov不等式

- $P(Y \geq t) \leq E[Y]/t$, 其中 Y 为非负随机变量, $t > 0$.





- 方差的性质与Chebyshev不等式

- 方差 $\sigma_x^2 = E[(X - \mu_x)^2]$, μ_x 为随机变量 X 的数学期望
- σ_x 称为标准差
- Chebyshev 不等式: $P(|X - \mu_x| > t\sigma_x) \leq 1/t^2$
- 如果随机变量 X 满足 $P(X=1)=p, P(X=0)=1-p$, 则 $\sigma_x^2 = p(1-p)$.
- 若 $X = \sum_{1 \leq i \leq n} X_i$, $\sigma_x^2 = \sum_{1 \leq i \leq n} \sigma_{x_i}^2$, X_i 是独立随机变量
- 若随机变量 X_i 满足 $P(X_i=1)=p, P(X_i=0)=1-p$, 则 $\sigma_x^2 = np(1-p)$.

• 算法的性能分析

定理. 算法执行1-9步一遍就可求出 $\min(S, k)$ 的概率是 $1-O(n^{-1/4})$, 即算法需要 $O(n)$ 次比较就可求出 $\min(S, k)$ 的概率是 $1-O(n^{-1/4})$.

证明. 若算法执行1-9一遍可求出 $\min(S, k)$, 则第6步需 $2n$ 次比较, 其他步需 $O(n)$ 次比较, 总需 $O(n)$ 次比较.

往证算法执行1-9一遍可求出 $\min(S, k)$ 的概率是 $1-O(n^{-1/4})$.

算法执行1-9一遍可求出 $\min(S, k)$ 的条件是:

- (1). $\min(S, k)$ 在 L 和 H 之间即 P 包含 $\min(S, k)$,
- (2). $|P| \leq 4n^{3/4} + 1$.

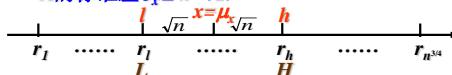
我们首先来计算上述两个条件失败的概率.

A. 计算条件(1)不成立的概率

条件(1)不成立当且仅当 $L > \min(S, k)$ 或 $H < \min(S, k)$.
令 $X_i=1$ 如果第 i 个随机样本 $\leq \min(S, k)$, 否则 $X_i=0$.

于是, $P(X_i=1)=k/n, P(X_i=0)=l-k/n$.

令 $X = \sum_{1 \leq i \leq n^{3/4}} X_i$ 是 R 中小于等于 $\min(S, k)$ 的样本数. 我们有 X 的数学期望 $\mu_x = n^{3/4}k/n = kn^{1/4}$, X 的方差 $\sigma_x^2 = n^{3/4}(k/n)(l-k/n) \leq n^{3/4}/4$, X 的标准差 $\sigma_x \leq n^{3/8}/2$.



如果 $L > \min(S, k), X < l$. 如果 $H < \min(S, k), X > h$. 于是
 $P(L > \min(S, k)) = P(X < l) = P(X < \mu_x - n^{1/2}) = P(|X - \mu_x| > n^{1/2})$,
 $P(H < \min(S, k)) = P(X > h) = P(X > \mu_x + n^{1/2}) = P(|X - \mu_x| > n^{1/2}) + (n^{3/4} + 1)$.
 应用 Chebyshev 不等式, 又由 $2n^{1/8} \sigma_x \leq n^{1/2}$, 我们有
 $P(|X - \mu_x| > n^{1/2}) \leq P(|X - \mu_x| > 2n^{1/8} \sigma_x) \leq 1/(2n^{1/8})^2 = O(n^{-1/4})$. 于是
 $P(L > \min(S, k)) = P(H < \min(S, k)) = O(n^{-1/4})$.

B. 计算 P 包含 $\min(S, k)$ 但 $|P| \leq 4n^{3/4} + 1$ 不成立的概率

令 $k_l = \min\{0, k - 2n^{3/4}\}, k_h = \max\{k + 2n^{3/4}, n\}$.

“ P 包含 $\min(S, k)$ 但 $|P| \leq 4n^{3/4} + 1$ 不成立” 发生当且仅当 $L < \min(S, k_l)$ 或 $H > \min(S, k_h)$.

类似与上面 A 中的分析,

$P(L < \min(S, k_l)) = P(H > \min(S, k_h)) = O(n^{-1/4})$.

由 A 和 B, “算法执行1-9一遍就可以求出 $\min(S, k)$ ” 不成立的概率是 $O(n^{-1/4})$.

即, “算法执行1-9一遍就可以求出 $\min(S, k)$ ” 的概率是 $1-O(n^{-1/4})$.



Las Vegas 算法

Las Vegas 算法(随机算法类 LV Algorithm)

- 算法不会产生不正确的解
- 算法一旦得到问题的解, 则该解是正确的
- 算法得到解的概率 $p > 0$
- 但算法运行过程可能不能产生问题的解
- 反复运行算法, 运行时间不确定, 最终可以得到问题的解
- 找到正确解需要运行算法的遍数与 p 相关(后续章节)
- 如: LazySelect 算法 1-9 步可视为一个 LV 算法
 - 一旦找到返回解, 该元素就是目标元素
 - 它运行一遍可能无法找到解
 - 反复运行, 最终必然得到问题的解
- 一般用来刻画 Yes-No 型问题的随机算法



• 比较3种中位数选择算法的性能

- 算法1: 排序后选择
- 算法2: 确定期中位数线性时间选择
第3章
- 算法3: 中位数选择随机算法

• 实验内容

- 实现3种算法
- 数据集寻找或生成
- 运行时间比较, 比较准确度(如何衡量)?
- 扩展性比较
- 以恰当、准确、规范地表述实验结果

实验



问题的定义

8.4 素数测试随机算法

- 问题的定义
- 随机算法设计
- 算法的性能分析
- 蒙特卡罗算法
- 简单的概率放大
- 蒙特卡罗 Vs 拉斯维加斯

- 输入
—一个正整数 N
- 输出
— N 是否素数



随机算法的设计

- 基本思想
 - 对 N 进行 m 次测试
 - 如果有一次测试成功，则回答 N 是合数
 - 如果 m 次测试均失败，则回答 N 是素数
 - 回答 N 是合数时，答案百分之百正确
 - 回答 N 是素数时，答案正确的概率是 $1-2^{-m}$

随机算法

1. 随机地选择 m 个数 $\{b_1, b_2, \dots, b_m\}$, 满足
 $1 \leq b_1, b_2, \dots, b_m \leq N$;
2. For $i=1$ To m Do
3. If $W(b_i)$ 成立 Then Return (N 是合数);
4. Return (N 是素数)

$W(b_i)$ 定义如下:

- (1) $b_i^{N-1} \neq 1 \pmod{N}$, 或
- (2) $\exists j [(N-1)/2^j = k$ 是整数, $1 < (b_i^k - 1)$ 与 N 的最大公因子 $< N$].



- 例1. 给定 $N=12$. 选择测试数集 $\{2, 3, 7\}$
- 测试 2: $2^{12-1} = 2048 \neq 1 \pmod{12}$, $W(2)$ 成立.
 N 是合数.

例2. 给定 $N=11$, 选择测试数集 $\{2, 5, 7\}$

测试 2: $2^{11-1} = 1024 \equiv 1 \pmod{11}$,

测试 5: $5^{11-1} = 9765625 \equiv 1 \pmod{11}$,

测试 7: $7^{11-1} = 282475249 \equiv 1 \pmod{11}$,

结论: 11 可能是素数

答案正确的概率为 $1-2^{-3}$



算法性能的分析

定理1. (1) 如果对于任意 $1 \leq b < N$, $W(b)$ 成立, 则 N 是合数.

(2) 如果 N 是合数, 则 $(N-1)/2 \leq |\{b \mid 1 \leq b < N, W(b)\}|$

*(1) 说明算法是正确的.

*(2) 说明, 如果 N 是合数, 则至少一半 $b (b < N)$ 使 $W(b)$ 成立

定理2. 算法的回答“ N 是素数”正确的概率是 $1 - 2^{-m}$.



蒙特卡罗算法

蒙特卡罗算法 (随机算法类 Monte Carlo Algorithm)

- 用于刻画 Yes-No 型计算问题
- 运行时间是固定的
- 算法得到正确解的概率 $p > 0$
- 算法得到错误解的概率 $1-p > 0$
- 单面错误蒙特卡罗算法 (MC1 算法)
 - > 算法输出 Yes 结论可靠
 - > 算法输出 No 结论可能是错的
- 如: N 是合数吗?
 - > Yes 可靠
 - > No 发生错误的可能性不超过 2^{-m}
- 双面错误蒙特卡罗算法 (MC2 算法)
 - > 算法输出 Yes 结论可能是错的
 - > 算法输出 No 结论可能是错的



MC1 算法的成功率放大

MC1 算法的成功率放大

- 用于刻画 Yes-No 型计算问题
- 运行时间是固定的
 - > Answer = Yes, 算法总输出 Yes
 - > Answer = No, $\Pr[\text{算法输出 No}] \geq \epsilon$
- 重复运行 t 次
 - > 如果 t 次均输出 Yes, 则最终输出 Yes
 - > 如果有一次输出 no, 则最终输出 no
 - > $\Pr[\text{算法犯错}] = \Pr[\text{answer} = \text{no} \text{ 但输出 Yes}] \leq (1-\epsilon)^t \leq \delta$
- 算法犯错的概率可以减小到任意指定的值 δ
- 可以建立重复遍数 t 与 ϵ, δ 之间的关系



MC2 算法的成功率放大

MC2 算法的成功率放大

- 用于刻画 Yes-No 型计算问题
- 运行时间是固定的
 - > Answer = Yes, $\Pr[\text{算法输出 Yes}] \geq 1/2 + \epsilon$
 - > Answer = No, $\Pr[\text{算法输出 No}] \geq 1/2 + \epsilon$
- 重复运行 t 次
 - > 输出占多数的答案

$$\Pr[\text{恰有 } i \text{ 次答案是正确的}] = \binom{t}{i} \left(\frac{1}{2} + \epsilon\right)^i \left(\frac{1}{2} - \epsilon\right)^{t-i}$$

$$\Pr[\text{算法最终犯错}] = \Pr[\text{正确次数} \leq t/2]$$

$$= \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} \left(\frac{1}{2} + \epsilon\right)^i \left(\frac{1}{2} - \epsilon\right)^{t-i} \leq \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} \left(\frac{1}{2} + \epsilon\right)^{t/2} \left(\frac{1}{2} - \epsilon\right)^{t/2}$$

$$= \left(\frac{1}{4} - \epsilon^2\right)^{t/2} \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} \leq \left(\frac{1}{4} - \epsilon^2\right)^{t/2} 2^t = (1 - 4\epsilon^2)^{t/2} = \delta$$



MC2 算法的成功率放大

- 用于刻画 Yes-No 型计算问题
- 运行时间是固定的
 - > Answer = Yes, $\Pr[\text{算法输出 Yes}] \geq 1/2 + \epsilon$
 - > Answer = No, $\Pr[\text{算法输出 No}] \geq 1/2 + \epsilon$
- 重复运行 t 次
 - > 输出占多数的答案

$$\Pr[\text{恰有 } i \text{ 次答案是正确的}] = \binom{t}{i} \left(\frac{1}{2} + \epsilon\right)^i \left(\frac{1}{2} - \epsilon\right)^{t-i}$$

$$\Pr[\text{算法最终犯错}] = \Pr[\text{正确次数} \leq t/2] \leq \delta$$

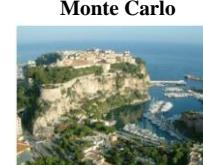
$$t = O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$$

- 算法犯错的概率可以减小到任意指定的值 δ
- 可以建立重复遍数 t 与 ϵ, δ 之间的关系



Monte Carlo

Las Vegas



蒙特卡罗 Vs 拉斯维加斯

两大类随机算法

Monte Carlo

Las Vegas



- 运行时间固定
- 是否正确是随机的
- 得到的解是正确的
- 也可能得不到解



由Las Vegas算法构造MC1算法

Tail Bound:

$$\Pr[X > t] < \epsilon$$

算法B(x)

1. 调用A(x)运行aT(n)步
2. 若A(x)获得解，则返回该解
3. 否则返回0

单面错误

Las Vegas算法→蒙特卡罗算法

- A是一个Las Vegas算法
- 最坏期望运行时间T(n)
- 得到的解是正确解

- B是一个蒙特卡罗算法

- 固定的运行时间aT(n)
- 如果得到解，则解是正确的
- 可能返回错误解

$$\begin{aligned} \Pr[B(x) \text{未获正确解}] &= \Pr[A(x) > aT(n)] \\ &< \frac{A(x) \text{的期望运行时间}}{aT(n)} \\ &= \frac{T(n)}{aT(n)} \\ &= \frac{1}{a} \end{aligned}$$



8.5 随机排序算法

- 问题的定义
- 随机算法
- 算法性能的分析
- 舍伍德算法



问题的定义



随机算法

- 输入
 - $S = \{x_1, x_2, \dots, x_n\}$
- 输出
 - 排序的S

基本思想

- 采用随机抽样的方法确定集合的划分点
- 把集合划分为两个子集合
- 分别递归地在每个子集合上使用随机排序算法



算法

1. 均匀等可能地在S中随机抽取一个样本y;
2. $\forall x \in S$ 与y比较，把S划分为如下两个集合：
 $S_1 = \{x | x \in S, x < y\}, \quad S_2 = \{x | x \in S, x > y\};$
3. 递归地排序 S_1 和 S_2 ;
4. 顺序输出排序的 S_1, y, S_2 ;



算法性能的分析

基本概念

- $S_{(i)}$ 表示S中阶为i的元素

例如， $S_{(1)}$ 和 $S_{(n)}$ 分别是最小和最大元素

- 随机变量 X_{ij} 定义如下：

$X_{ij}=1$ 如果 $S_{(i)}$ 和 $S_{(j)}$ 在运行中被比较，否则为0

- X_{ij} 是 $S_{(i)}$ 和 $S_{(j)}$ 的比较次数

- 算法的比较次数为 $\sum_{i=1}^n \sum_{j>i} X_{ij}$

- 算法的平均复杂性为 $E[\sum_{i=1}^n \sum_{j>i} X_{ij}] = \sum_{i=1}^n \sum_{j>i} E[X_{ij}]$



HIT

CS&E

• 计算 $E[X_{ij}]$

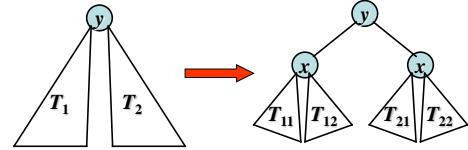
- 设 p_{ij} 为 $S_{(i)}$ 和 $S_{(j)}$ 在运行中比较的概率，则

$$E[X_{ij}] = p_{ij} \times 1 + (1-p_{ij}) \times 0 = p_{ij}$$

关键问题成为求解 p_{ij}

• 求解 P_{ij}

- 我们可以用树表示算法的计算过程



- 我们可以观察到如下事实：

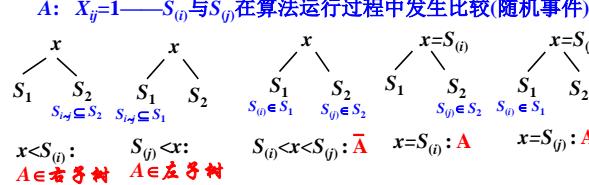
- 一个子树的根必须与其子树的所有节点比较
- 不同子树中的节点不可能比较
- 任意两个节点至多比较一次



HIT

CS&E

$$S_{i,j} = \{S_{(i)}, S_{(i+1)}, S_{(i+2)}, \dots, S_{(j)}\}$$



计算 p_{ij}



综上所述

$$\begin{aligned} E[T(n)] &= E[\sum_{i=1}^n \sum_{j>i} X_{ij}] = \sum_{i=1}^n \sum_{j>i} E[X_{ij}] = \sum_{i=1}^n \sum_{j>i} p_{ij} \\ &= \sum_{i=1}^n \sum_{j>i} \frac{2}{j-i+1} = \sum_{i=1}^{n-i+1} \frac{2}{k} \leq \sum_{i=1}^n \sum_{k=1}^n \frac{2}{k} \\ &\leq 2 \sum_{i=1}^n \sum_{k=1}^n \frac{1}{k} = 2n \sum_{k=1}^n \frac{1}{k} \\ &= O(n \log n) \end{aligned}$$

定理. 随机排序算法的期望时间复杂性为 $O(n \log n)$



HIT

CS&E

舍伍德算法



HIT

CS&E

舍伍德算法(随机算法类 Sherwood Algorithm)

- 确定型算法的随机化
- 消除算法在最好实例和最坏实例间差别
- 如：QuickSort算法也可以确定性地选择划分元素
 - 最好时间复杂度为 $O(n \log n)$
 - 最坏时间复杂度为 $O(n^2)$
 - 随机选择划分子元素后，期望时间复杂度为 $O(n \log n)$
 - 先设计了确定型算法，随机化之后才得QuickSort
- 舍伍德算法总能得到问题的正确解

8.6 最小割随机算法

- 问题定义
- 随机算法
- 算法性能的分析
- 成功概率放大技术



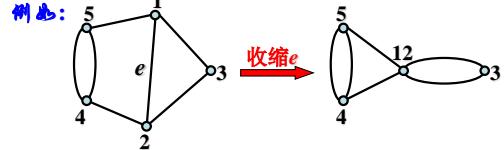
问题定义

- 输入:
 - 无向多重连通图 G
 - 输出
 - 一个 Min-Cut
- 图 G 的一个 Cut 是一组边, 从 G 中删除这个 Cut 将导致两个或多个连通分量
- Cut 的大小是其边数, 多重边重复计算
- 最小 Cut 是具有最少边的 Cut



基本概念

- Cut 可以视简单连通集的划分 $V = (C, V - C)$, Cut 是所有 G 中连接 C 和 $V - C$ 的边集合.
- 图 G 的边 (x, y) 的 contraction:
 - 用新节点代替节点 x 和 y 或边 (x, y) ,
 - $\forall v \in V$, 用边 (v, z) 替换边 (x, v) 或 (y, v) ,
 - G 的其余部分保持不变



随机算法

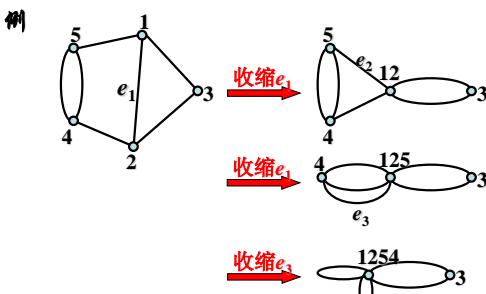


- 我们用 $G/(x, y)$ 表示 G 的边 (x, y) 的收缩
- 边集合 $F \subseteq G$ 收缩记作 G/F
- 图 G/F 的节点集合表示为 V/F
- 图 G/F 的节点集合表示为 E/F



随机算法 CONTRACTION

1. $H = G$;
2. While $|H(V)| > 2$ Do
3. 随机地从 $H(E)$ 中选择一条边 (x, y) ;
4. $F = F \cup \{(x, y)\}$;
5. $H = H/(x, y)$;
6. Cut = 连接 H 中两个元节点的 G 的所有边.



$Cut = \{(1,3), (2,3)\}$



算法的性能分析

定理1. 如果算法的输入是具有 n 个节点的多重图, 则
算法的时间复杂性为 $O(n^2)$.

证明. 一次边收缩需要 $O(n)$ 时间.

至多进行 $O(n)$ 次收缩.

于是, 算法时间复杂性为 $O(n^2)$.

注意:

我们仅证明了在 $O(n^2)$ 时间内算法能够求出一个 Cut,
但是这个 Cut 不一定是优化的.



- 引理1.** 如果 k 是min-cut的大小, 则 G 至少有 $kn/2$ 条边。
证. 如果 $|G(E)| < kn/2$, 则存在一个度小于 k 的节点 p .
 删除与 p 相关连的 k 条, 把 G 划分为两个连通分量, 其一是仅包含 p .
 于是, 与 p 相关连的边集合是一个cut.
 但是这个cut的大小 $< k$, 与min-cut大小为 k 矛盾.
引理2. 算法输出的cut是连接两个剩余节点的没有被收缩过的边。
证. 从算法定义可以看到, 算法输出的cut是连接两个剩余节点的没有被收缩过的边的集合。



引理3. 设图 G 的min-cut的大小为 k , 则 $G/(u,v)$ 是 G 收缩边 (u,v) 后得到的图。如果 (u,v) 不是最小割中的边, 则 $G/(u,v)$ 的最小割至少为 k .

证. 反证法. 记 $G/(u,v)$ 的最小割为 C , 且 $|C| < k$



从 $G/(u,v)$ 删除 C 中的边, 得到两个顶点子集 $V_1, V_2, uv \in V_2$
 从 G 删除 C 中的边, 得顶点子集 $V_1 = V_2 - \{uv\} \cup \{u, v\}$
 C 也是 G 的割, 且 $|C| < k$. 矛盾

- 定理2.** 设 C 是一个min-cut, 其大小为 k . 在算法结束时, C 中无边被收缩过的概率大于 $2/n^2$ 。
证. A_i 表示第 i 步没有选中 C 的边, $1 \leq i \leq n-2$.
 在第1步, 选中的边在 C 中的概率至多为 $k/(kn/2) = 2/n$, 即
 $\Pr(A_1) \geq 1 - 2/n$.
 在第2步, 若 A_1 发生, 则至少有 $k(n-1)/2$ 条边(每次收缩减少一个节点), 选中 C 中边的概率为 $2/(n-1)$, 即
 $\Pr(A_2 | A_1) \geq 1 - 2/(n-1)$.
 在第 i 步, 若 A_1 至 A_{i-1} 发生, 则有 $n-i+1$ 个节点, 即至少有 $k(n-i+1)/2$ 条边, 于是
 $\Pr(A_i | \cap_{1 \leq j \leq i-1} A_j) \geq 1 - 2/(n-i+1)$
 最后我们有
 $\Pr(\cap_{1 \leq i \leq n-2} A_i) \geq \prod_{1 \leq i \leq n-2} (1 - 2/(n-i+1)) = 2/n(n-1) > 2/n^2$



算法Amplify

放大成功率: 简单重复

1. $S = E$
2. For $i=1$ to n^2 Do
3. $S_i = \text{CONTRACTION}(G)$;
4. If $|S| > |S_i|$ Then $S = S_i$;
5. Return S

推论1. 算法Amplify的运行时间为 $O(n^4)$, 它不能发现一个min-cut的概率为

$$\left(1 - \frac{2}{n^2}\right)^{n^2/2} < \frac{1}{e}$$



算法 Amplify

1. $S = E$
2. For $i=1$ to n^2 Do
3. $S_i = \text{CONTRACTION}(G)$;
4. If $|S| > |S_i|$ Then $S = S_i$;
5. Return S

这有意义吗? 🤔

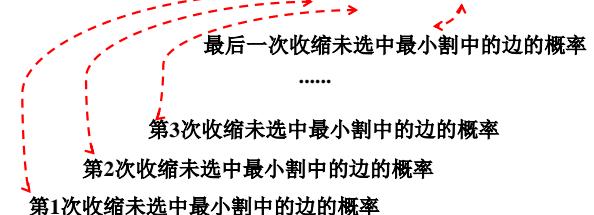
最小割问题的精确算法的时间复杂度是多少? $O(n^3)$

造成这种现象的原因是什么呢? 获得正确解的概率太低
 怎么才能提高获得正确解的概率呢?



成功率低下的原因

$$\Pr[\text{CONTRACT成功}] \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} \geq \frac{1}{n^2}$$



.....

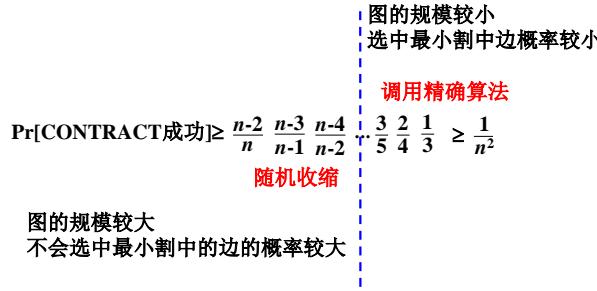
第3次收缩未选中最小割中的边的概率

第2次收缩未选中最小割中的边的概率

第1次收缩未选中最小割中的边的概率



成功率放大：重复关键操作



算法DetRan

1. While $|V| > d(n)$ Do // $d(n)=n^{2/3}$
2. 随机选择一条边进行收缩;
3. 调用精确算法求得最小割 S ;
4. Return S

第2步每次时间复杂度为 $O(n)$ 执行 $n-d(n)$ 遍

第3步每次时间复杂度为 $O(d^3(n))$

总时间为 $O(n^2) + O(d^3(n))$

结论：取 $d(n)=n^{2/3}$, 算法DetRan的时间复杂度为 $O(n^2)$



算法DetRan

1. While $|V| > d(n)$ Do
2. 随机选择一条边进行收缩;
3. 调用精确算法求得最小割 S ;
4. Return S

$$\begin{aligned} \Pr[\text{DetRan获得最小割}] &\geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{d(n)+1}{d(n)+3} \cdot \frac{d(n)}{d(n)+2} \cdot \frac{d(n)-1}{d(n)+1} \\ &= \frac{d(n)}{n} \cdot \frac{d(n)-1}{n-1} \\ &\approx \frac{n^{4/3}}{n^2} \quad d(n) = n^{2/3} \\ &= \frac{1}{n^{2/3}} \end{aligned}$$



算法Amplify2

1. $S=E$
2. For $i=1$ to $n^{2/3}$ Do
3. $S_i = \text{DetRan}(G)$;
4. If $|S| > |S_i|$ Then $S = S_i$;
5. Return S

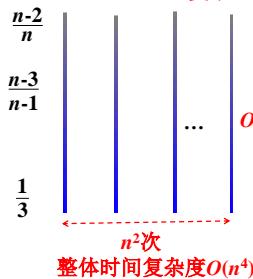
$$\Pr[\text{Amplify2未得最小割}] \leq \left[1 - \frac{1}{n^{2/3}}\right]^{n^{2/3}} \approx e^{-1}$$

结论：独立运行DetRan算法 $n^{2/3}$ 遍, 时间复杂度为 $O(n^{8/3})$
找到最小割的概率至少为 $1-e^{-1}$

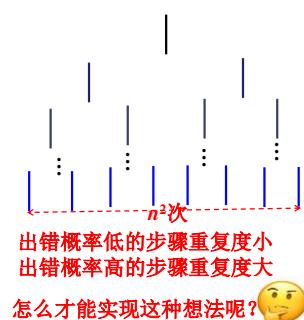


成功率放大：关键操作重复策略

CONTRACT重复示意



CONTRACT重复新思路



算法RepTree(G)

//顶点个数记为 n

1. If $n \leq 6$ Then 用确定型算法求最小割 S , 返回
2. $h = \lceil n \cdot n/2^{1/2} \rceil$;
3. 随机独立收缩 G 中 $n-h$ 条边得图 G_1 ; $O(n^2)$
4. 随机独立收缩 G 中 $n-h$ 条边得图 G_2 ; $O(n^2)$
5. $S_1 = \text{RepTree}(G_1)$;
6. $S_2 = \text{RepTree}(G_2)$;
7. Return $\min(S_1, S_2)$;

$$T(n) = 2T(n/2^{1/2}) + O(n^2)$$

$$T(n) = O(n^2 \log n)$$

算法RepTree(G)

//顶点个数记为 n

1. If $n \leq 6$ Then 用确定型算法求最小割 S ,返回
2. $h = \lceil n - n/2^{1/2} \rceil$;
3. 随机独立收缩 G 中 $n-h$ 条边得图 G_1 ;
4. 随机独立收缩 G 中 $n-h$ 条边得图 G_2 ;
5. $S_1 = \text{RepTree}(G_1)$;
6. $S_2 = \text{RepTree}(G_2)$;
7. Return $\min(S_1, S_2)$;

$$\Pr[G_1 \text{仍含最小割}] \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{h+1}{h+3} \cdot \frac{h}{h+2} \cdot \frac{h-1}{h+1}$$

$$= \frac{h(h-1)}{n(n-1)} \geq 1/2$$



RepTree获得正确解的概率

$\Pr(n) = \text{算法在规模为 } n \text{ 的图上获得正确解的概率}$

$\Pr(n/2^{1/2}) = \text{算法在 } G_1 \text{ 上获得正确解的概率} | G_1 \text{ 含最小割}$

$\Pr(n/2^{1/2}) = \text{算法在 } G_2 \text{ 上获得正确解的概率} | G_2 \text{ 含最小割}$

$\Pr[G_1, G_2 \text{ 均不包含最小割}] \leq (1-1/2)(1-1/2) = 1/4$

$\Pr[\text{算法找不到正确解} | G_1, G_2 \text{ 含最小割}]$

$\leq \Pr[\text{未找到正确解} | G_1 \text{ 含最小割}] \cdot \Pr[\text{未找到正确解} | G_2 \text{ 含最小割}]$

$= [1 - \Pr(n/2^{1/2})]^2$

$$\Pr(n) = 1 - \Pr[\text{算法无法获得正确解的概率}]$$

$$\geq 3/4 - [1 - \Pr(n/2^{1/2})]^2$$

解得: $\Pr(n) = \Omega(1/\log n)$



结论



结论: 算法RepTree的时间复杂度为 $O(n^2 \log n)$

找到最小割的概率至少为 $\Omega(1/\log n)$

重复运行RepTree算法 $\log n$ 遍的时间开销为
 $O(n^2 \log^2 n)$, 找到正确解的概率为 $1-e^{-1}$

重复运行RepTree算法 $\log^2 n$ 遍的时间开销为
 $O(n^2 \log^3 n)$, 找到正确解的概率接近于1

8.7 附录—概率基础



概率空间

样本空间 Ω : 所有基本事件（也称为样本）构成的集合

事件集合 Σ : Ω 的一个子集称为一个事件

(K1): $\emptyset, \Omega \in \Sigma$ **\emptyset -不可能事件, Ω -必然事件**

(K2): \cup, \cap, \setminus 下 Σ 封闭 **Σ 是 σ -代数**

概率测度 \Pr : $\Sigma \rightarrow R$ 取非负值

(K3): $\Pr(\Omega) = 1$

(K4): $A \cap B = \emptyset \Rightarrow \Pr(A \cup B) = \Pr(A) + \Pr(B)$

(K5*): $A_1 \supset A_2 \supset \dots \text{且} \cap_n A_n = \emptyset \Rightarrow \lim_{n \rightarrow \infty} \Pr(A_n) = 0$



事件:

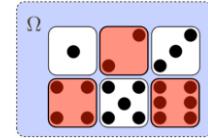
谓词 \mathcal{E}

集合 $A = \{x \in \Omega | \mathcal{E}(x)\}$

概率:

$\Pr[\mathcal{E}] = \Pr[A]$

$$\begin{array}{ll} \vee & \cup \\ \wedge & \cap \\ \neg & \Omega \setminus \\ \Rightarrow & \subseteq \end{array}$$



\mathcal{E} = 掷出偶数点

$$\begin{aligned} \Pr[\mathcal{E}] &= \Pr[A] \\ &= \Pr[\square \vee \square \vee \square] \\ &= \Pr[\square] + \Pr[\square] + \Pr[\square] \end{aligned}$$



HIT
CS&E

- (K1): $\emptyset, \Omega \in \Sigma$
- (K2): \cup, \cap, \setminus 下 Σ 封闭
- (K3): $\Pr(\Omega) = 1$
- (K4): $A \cap B = \emptyset \Rightarrow \Pr(A \cup B) = \Pr(A) + \Pr(B)$

$$\Pr[\neg E] = 1 - \Pr[E] \quad \neg E \vee E = \Omega \Rightarrow \Pr[\neg E] + \Pr[E] = 1$$

$$E_1 \Rightarrow E_2, \text{ 则 } \Pr[E_1] \leq \Pr[E_2] \quad \Pr[E_2] = \Pr[E_1] + \Pr[E_2 \wedge \neg E_1]$$

$$\Pr[E_1 \vee E_2] = \Pr[E_1] + \Pr[E_2] - \Pr[E_1 \wedge E_2]$$



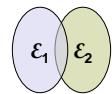
HIT
CS&E

$$\Pr[E_1 \vee E_2] = \Pr[E_1] + \Pr[E_2] - \Pr[E_1 \wedge E_2]$$

$$\Pr[E_1] = \Pr[E_1 \wedge \neg(E_1 \wedge E_2)] + \Pr[E_1 \wedge E_2]$$

$$\Pr[E_2] = \Pr[E_2 \wedge \neg(E_1 \wedge E_2)] + \Pr[E_1 \wedge E_2]$$

$$\Pr[E_1 \vee E_2] = \Pr[E_1 \wedge \neg(E_1 \wedge E_2)] + \Pr[E_2 \wedge \neg(E_1 \wedge E_2)] + \Pr[E_1 \wedge E_2]$$



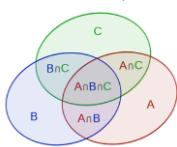
将前两式带入第三式，消除无关项，即得结论



HIT
CS&E

$$\Pr[E_1 \vee E_2] = \Pr[E_1] + \Pr[E_2] - \Pr[E_1 \wedge E_2]$$

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$$



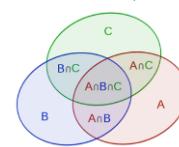
$$\left| \bigcup_{1 \leq i \leq n} S_i \right| = \sum_{i=1}^n |S_i| - \sum_{A \in \binom{[n]}{2}} \left| \bigcap_{i \in A} S_i \right| + \sum_{A \in \binom{[n]}{3}} \left| \bigcap_{i \in A} S_i \right| - \sum_{A \in \binom{[n]}{4}} \left| \bigcap_{i \in A} S_i \right| - \cdots + (-1)^{k-1} \sum_{A \in \binom{[n]}{k}} \left| \bigcap_{i \in A} S_i \right| + \cdots + (-1)^{n-1} \left| \bigcap_{i=1}^n S_i \right|.$$



HIT
CS&E

$$\Pr[E_1 \vee E_2] = \Pr[E_1] + \Pr[E_2] - \Pr[E_1 \wedge E_2]$$

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$$



$$\Pr \left[\bigvee_{1 \leq i \leq n} \mathcal{E}_i \right] = \sum_{i=1}^n \Pr[\mathcal{E}_i] - \sum_{\binom{A}{2} \subseteq [n]} \Pr \left[\bigwedge_{i \in A} \mathcal{E}_i \right] + \sum_{\binom{A}{3} \subseteq [n]} \Pr \left[\bigwedge_{i \in A} \mathcal{E}_i \right] - \sum_{\binom{A}{4} \subseteq [n]} \Pr \left[\bigwedge_{i \in A} \mathcal{E}_i \right] - \cdots + (-1)^{k-1} \sum_{\binom{A}{k} \subseteq [n]} \Pr \left[\bigwedge_{i \in A} \mathcal{E}_i \right] + \cdots + (-1)^{n-1} \Pr \left[\bigwedge_{i=1}^n \mathcal{E}_i \right].$$



HIT
CS&E

合并界限(The Union Bound)

$$\Pr[\mathcal{E}_1 \vee \mathcal{E}_2] = \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2] - \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2]$$

$$\Pr[\mathcal{E}_1 \vee \mathcal{E}_2] \leq \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2]$$

数学归纳

$$\Pr \left[\bigvee_{1 \leq i \leq n} \mathcal{E}_i \right] \leq \sum_{i=1}^n \Pr[\mathcal{E}_i]$$

适用于任何依赖关系！



HIT
CS&E

合并界限(续)

$$\Pr[\text{GOOD}_1] = 1 - \varepsilon_1$$

$$\Pr[\text{GOOD}_2] = 1 - \varepsilon_2$$

~~$$\Pr[\text{GOOD}_1 \wedge \text{GOOD}_2] =$$~~

~~$$\Pr[\text{GOOD}_1] \cdot \Pr[\text{GOOD}_2]$$~~

二者可能不是独立的

$$\begin{aligned} \Pr[\text{GOOD}_1 \wedge \text{GOOD}_2] &= 1 - \Pr[\text{BAD}_1 \vee \text{BAD}_2] \\ &\geq 1 - (\Pr[\text{BAD}_1] + \Pr[\text{BAD}_2]) \\ &= 1 - \varepsilon_1 - \varepsilon_2 \end{aligned}$$

使用过程中涉及误差累积



条件概率

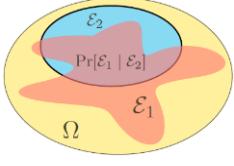
定义: “在随机事件 \mathcal{E}_2 发生的条件下 \mathcal{E}_1 发生”的**条件概率**

概率定义为

$$\Pr[\mathcal{E}_1 | \mathcal{E}_2] = \frac{\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2]}{\Pr[\mathcal{E}_2]}$$

如果 \mathcal{E}_1 和 \mathcal{E}_2 是独立的

$$\begin{aligned}\Pr[\mathcal{E}_1 | \mathcal{E}_2] &= \frac{\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2]}{\Pr[\mathcal{E}_2]} \\ &= \frac{\Pr[\mathcal{E}_1] \Pr[\mathcal{E}_2]}{\Pr[\mathcal{E}_2]} \\ &= \Pr[\mathcal{E}_1]\end{aligned}$$



条件概率链

$$\Pr[\mathcal{E}_1 | \mathcal{E}_2] = \frac{\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2]}{\Pr[\mathcal{E}_2]} \longrightarrow \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2] = \Pr[\mathcal{E}_1 | \mathcal{E}_2] \cdot \Pr[\mathcal{E}_2]$$

$$\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \dots \wedge \mathcal{E}_n] = \Pr[(\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \dots \wedge \mathcal{E}_{n-1}) \wedge \mathcal{E}_n]$$

$$= \Pr[\mathcal{E}_n / \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \dots \wedge \mathcal{E}_{n-1}] \cdot \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \dots \wedge \mathcal{E}_{n-1}] \\ = \dots \quad (\text{在 } \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \dots \wedge \mathcal{E}_{n-1} \text{ 重复上述过程})$$

$$= \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdot \Pr[\mathcal{E}_3 | \mathcal{E}_1 \wedge \mathcal{E}_2] \cdot \dots \cdot \Pr[\mathcal{E}_n | \mathcal{E}_1 \wedge \dots \wedge \mathcal{E}_{n-1}]$$

$$\text{对任意 } \mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n \text{ 有: } \Pr\left[\bigwedge_{i=1}^n \mathcal{E}_i\right] = \prod_{k=1}^n \Pr\left[\mathcal{E}_k \mid \bigwedge_{i < k} \mathcal{E}_i\right]$$

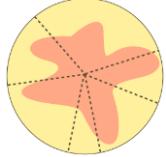


全概率公式

若 Ω 被划分为 $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n$ （即 $\mathcal{E}_i \wedge \mathcal{E}_j = \emptyset, \forall i \neq j$, $\bigcup_i \mathcal{E}_i = \Omega$ ），则

$$\Pr[\mathcal{E}] = \sum_{i=1}^n \Pr[\mathcal{E} \wedge \mathcal{E}_i] = \sum_{i=1}^n \Pr[\mathcal{E} | \mathcal{E}_i] \cdot \Pr[\mathcal{E}_i]$$

对任意 \mathcal{E} 成立



全概率公式常用于
分情况讨论事件的概率

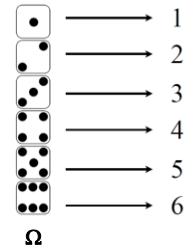


随机变量

概率空间: (Ω, Σ, \Pr)

投掷结果是一个随机变量

随机变量X: 一个函数 $X: \Omega \rightarrow R$



随机变量

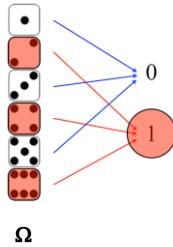
概率空间: (Ω, Σ, \Pr)

投掷结果的奇偶是一个随机变量

随机变量X: 一个函数 $X: \Omega \rightarrow R$

事件 “ $X=x$ ” 即 “ $\{s \in \Omega | X(s)=x\}$ ”

$$\Pr[X=x] = \Pr[\{s \in \Omega | X(s)=x\}]$$



随机变量的独立性

定义: 随机变量 X 和 Y 是**独立**的，如果

$$\Pr[X=x \wedge Y=y] = \Pr[X=x] \cdot \Pr[Y=y]$$

对任意 x, y 成立

随机变量 X_1, X_2, \dots, X_n 是**相互独立**的，如果

$$\Pr[\bigwedge_{i \in I} (X_i = x_i)] = \prod_{i \in I} \Pr[X_i = x_i]$$

对任意 $I \subseteq [n]$ 和任意 $x_i (i \in I)$ 均成立



随机变量的数学期望

定义. 离散随机变量 X 的数学期望定义为

$$E[X] = \sum_x x \cdot \Pr[X=x]$$

其中 x 取遍 X 的值域

期望的线性性质

$$E\left[\sum_{i=1}^n a_i X_i\right] = \sum_{i=1}^n a_i \cdot E[X_i]$$



期望的线性性质

$$E\left[\sum_{i=1}^n a_i X_i\right] = \sum_{i=1}^n a_i \cdot E[X_i]$$

证明:

$$\begin{aligned} E[X+Y] &= \sum_x \sum_y (x+y) \Pr[X=x \wedge Y=y] \\ &= \sum_x \sum_y x \Pr[X=x \wedge Y=y] + \sum_x \sum_y y \Pr[X=x \wedge Y=y] \\ &= \sum_x x \sum_y \Pr[X=x \wedge Y=y] + \sum_y y \sum_x \Pr[X=x \wedge Y=y] \\ &\stackrel{\text{全概率公式}}{=} \sum_x x \Pr[X=x] + \sum_y y \Pr[Y=y] \\ &= E[X] + E[Y] \end{aligned}$$



期望的线性性质

$$E\left[\sum_{i=1}^n a_i X_i\right] = \sum_{i=1}^n a_i \cdot E[X_i]$$

证明:

$$E[X+Y] = E[X] + E[Y]$$

$$\begin{aligned} E[cX] &= \sum_x x \Pr[cX=x] \\ &= c \sum_x \frac{x}{c} \Pr[X=\frac{x}{c}] \\ &= c \sum_{x'} x' \Pr[X=x'] \\ &= cE[X] \end{aligned}$$



期望的线性性质

$$E\left[\sum_{i=1}^n a_i X_i\right] = \sum_{i=1}^n a_i \cdot E[X_i]$$

证明:

$$E[X+Y] = E[X] + E[Y]$$

$$E[cX] = cE[X]$$

在此基础上, 对 n 做数学归纳, 得出结论

注: 证明过程未涉及变量间是否独立
线性性质对任何依赖关系都成立



该性质的应用实例



proof

猴子在打字机上随机连续地敲出一个长度为10亿的字符串
这个字符串中“proof”平均出现多少次呢?

$X_i=1$ —— “proof”出现在位置 i

$$\Pr[X_i=1] = 1/26^5$$

$X_i=0$ —— “proof”未出现在位置 i

$$\Pr[X_i=0] = 1 - 1/26^5$$

$X = \sum_i X_i$ —— “proof”出现的总次数

$$E[X_i] = 1/26^5$$

$$E[X] = E\left[\sum_{i=1}^{10^9 \cdot 4} X_i\right] = \sum_{i=1}^{10^9 \cdot 4} E[X_i] = (10^9 \cdot 4)E[X_1] = (10^9 \cdot 4)/26^5 \approx 84$$



对于任意非负随机变量 X ,

$$\Pr[X \geq t] \leq \frac{E[X]}{t}$$

对任意 $t>0$ 成立

$$\begin{aligned} \text{证明: } \Pr[X \geq t] &= \int_t^{+\infty} \Pr[X=x] dx \\ &= \int_t^{+\infty} 1 \cdot \Pr[X=x] dx \\ &\leq \int_t^{+\infty} \frac{x}{t} \cdot \Pr[X=x] dx \\ &= \frac{1}{t} \int_t^{+\infty} x \cdot \Pr[X=x] dx \\ &\leq \frac{1}{t} \int_0^{+\infty} x \cdot \Pr[X=x] dx \\ &= \frac{E[X]}{t} \end{aligned}$$

Markov不等式





Markov不等式

对于任意**非负**随机变量 X ,
 $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$

对任意 $t > 0$ 成立

推广

对任意随机变量 X 和**非负**函数 $h: X \rightarrow R^+$
 $\Pr[h(X) \geq t] \leq \frac{\mathbb{E}[h(X)]}{t}$

对任意 $t > 0$ 成立



方差

定义. 离散随机变量 X 的方差定义为

$$\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

方差的算术平方根称为 X 的标准差, 记为 $\delta[X]$

$$\delta[X] = (\text{Var}[X])^{1/2}$$

$$\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

$$= \mathbb{E}[X^2 - 2X \cdot \mathbb{E}[X] + (\mathbb{E}[X])^2]$$

$$= \mathbb{E}[X^2] - 2\mathbb{E}[X] \cdot \mathbb{E}[X] + (\mathbb{E}[X])^2$$

$$= \mathbb{E}[X^2] - 2\mathbb{E}[X] \cdot \mathbb{E}[X] + (\mathbb{E}[X])^2$$

$$= \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$



二项分布的方差

定义. 若随机变量 X 满足

$$\Pr[X=1] = p \quad \Pr[X=0] = 1-p$$

则称 X 服从**两点分布**



Chebyshev不等式

对任意随机变量 X ,

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\text{Var}[X]}{t^2}$$

对任意 $t > 0$ 成立

证明: 令 $Y = (X - \mathbb{E}[X])^2$, Y 是非负随机变量

$$|X - \mathbb{E}[X]| \geq t \Leftrightarrow Y \geq t^2$$

对 Y 和 t^2 运用Markov不等式即得结论

$$\mathbb{E}[X] = 1 \cdot \Pr[X=1] + 0 \cdot \Pr[X=0] = p$$

$$\begin{aligned} \text{Var}[X] &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= (1-p)^2 \cdot \Pr[X=1] + (0-p)^2 \cdot \Pr[X=0] \\ &= (1-p)^2 \cdot p + p^2 \cdot (1-p) \\ &= p(1-p) \end{aligned}$$



对任意随机变量 X ,
 $\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\text{Var}[X]}{t^2}$

对任意 $t > 0$ 成立



对任意随机变量 X ,
 $\Pr[|X - \mathbb{E}[X]| \geq t \cdot \delta[X]] \leq \frac{1}{t^2}$

对任意 $t > 0$ 成立



尾概率界(Tail Bound)

Tail Bound: $\Pr[X > t] < \epsilon$

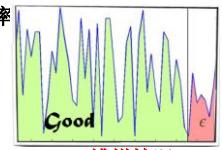


理想情况

很多分析任务要求给出尾概率的界限

如:

- (1) 随机算法运行时间超过阈值的概率
- (2) 某种代价 (如最大负载)
- (3) 极端情况出现的概率



糟糕情况



Tail Bound: $\Pr[X > t] < \epsilon$

分析手段1:
计数并计算概率
涉及大量计算
涉及计算方法的有效性

尾概率界(Tail Bound)

将 n 个球放进 n 个箱子:

$$\begin{aligned} \Pr[\text{第一个箱子内球个数} > t] &\leq \binom{n}{t} \left(\frac{1}{n}\right)^t \\ &= \frac{n!}{t!(n-t)!n^t} \\ &= \frac{1}{t!} \cdot \frac{n(n-1)(n-2)\cdots(n-t+1)}{\prod_{i=0}^{t-1} \left(1 - \frac{i}{n}\right)} n^t \\ &\leq \frac{1}{t!} \\ &\leq \left(\frac{e}{t}\right)^t \end{aligned}$$



Tail Bound: $\Pr[X > t] < \epsilon$

分析手段2:
将尾概率的计算归结
为分析 X 某种特征 I

分析工具:
Markov不等式(仅知 $E[X]$ 是紧的)
Chebyshev不等式
Chernoff界 (待讲)
其他概率不等式 (待讲)

尾概率界(Tail Bound)

X 服从分布 \mathcal{D}



Tail Bound: $\Pr[X > t] < \epsilon$

分析手段2:
将尾概率的计算归结
为分析 X 某种特征 I

尾概率界(Tail Bound)

将 n 个球放进 n 个箱子:

$$\begin{aligned} X_i=1 &\text{ 第 } i \text{ 个球落入第一个箱子} \\ X_i=0 &\text{ 第 } i \text{ 个球未落入第一个箱子} \\ X = \sum_i X_i &\text{ 第一个箱子中球总数} \\ \Pr[X_i=1] = 1/n &\quad \Pr[X_i=0] = 1-1/n \\ E[X_i] = 1/n & \\ E[X] = \sum_i E[X_i] = 1 & \\ \Pr[\text{第一个箱子内球个数} > t] &= \Pr[X > t] \\ &\leq E[X]/t \\ &= 1/t \end{aligned}$$

你试试Chebyshev不等式呢?