随机算法 (简介)

算法设计与分析小班课

2022 年 6 月

注意: 以下内容主要是给课本作一些注解并补充习题, 可能不能准确地代表课程内容和考核的方向, 也不能替代阅读课本、听课、完成作业题、做往年题等活动; 其中的内容没有经过课程主管老师的审核, 也可能存在错误. 不过出现的所有错误都由作者本人负责.

由于课本上的随机算法部分的内容比较少, 而后续还有专门的「随机算法」课程 (春季学期 04834010, 孔雨晴老师), 所以以下浅尝辄止.

虽然随机性是 "自然" 的, 但是为什么引入随机性可能有助于改进算法? 简单的理由如下: 因为随机能让算法以一定的概率避开糟糕的东西, 或者以一定的概率 "撞大运"; 前者相当于说攻击确定性算法的最坏输入方法不再 (直接地) 适用于随机算法, 后者相当于说在搜索空间很大或者 "遇事不决" 的时候, 随机能帮助算法作一些合理的猜测.

在随机算法的设计与分析中需要用到概率论知识. 其中比较重要的有 Bool 不等式 (a.k.a. union bound)、条件概率与条件期望、期望的线性性、集中不等式 (concentration inequalities, 例如 Chernoff 界) 和随机过程, 可从 [MRT18, 附录 C 和 D] 中找到介绍.

1 几个新例子

下面分别给出分析中用到条件概率、期望的线性性和 Chernoff 界的例子.

例 1 (Karger, 1993) 这是一个很有名的针对最小割的随机算法 [Kar93], 这里考虑的是无向图, 每条边的边权都是 1. 算法如下:

(i) 每次随机选一条边, 收缩之. 这里, 收缩边后产生的重边我们都保留.

(ii) 当收缩得到的图只有两个点时, 输出这两个顶点之间的全部边作为最小割.

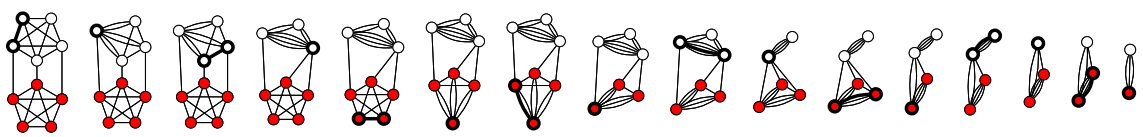

图 1: Karger 算法的一次成功执行过程. 图片来源: Wikimedia Commons, 使用许可: CC BY-SA 3.0

设所说的图 G 有 n 个顶点, 上面算法至多执行 $n-2$ 步. 它输出割是得到保证的, 因为收缩边得到的新图中的割仍然是原图中的割.

让我们来分析这个算法真的输出最小割的概率. 设 k 是最小割的大小. 一条边被收缩意味着它被排除在最后的输出之外, 我们用 A_i 表示事件: 在算法的第 i 步没有扔掉最小割中的一条边. 根据握手定理我们知道图中至少有 $\frac{kn}{2}$ 条边, 否则存在一个点的度小于 k, 所以 $\Pr[A_1] \geq 1 - \frac{k}{\frac{kn}{2}} = 1 - \frac{2}{n}$. 进一步,

$$\Pr[A_2 \mid A_1] \geq 1 - \frac{2}{n-1}, \ldots, \Pr\left[A_i \,\middle|\, \bigcap_{j=1}^{i-1} A_j\right] \geq 1 - \frac{2}{n-i+1}.$$

根据条件概率的乘法公式我们有

$$\Pr[A_1 A_2 \cdots A_{n-2}] \geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \frac{2}{n(n-1)} = \Omega(n^{-2}).$$

当我们重复执行该算法 $\Omega(n^2)$ 次, 选择得到的割中最小的, 答案错误的概率就不超过 $\left(1 - \frac{2}{n(n-1)}\right)^{n^2} < \frac{1}{\mathrm{e}}$; 再进一步执行, 则错误概率可以任意小. (算法的具体运行时间和收缩边的实现方法有关.) ◇

例 2 MAX-3SAT 问题: 给一个有 n 个变元和 k 个子句的 3-CNF, 求一个变元的赋值, 使得被满足的子句个数尽可能多.

随机算法: 对于每个变元 x_i, 我们分别独立地以 $\frac{1}{2}$ 的概率给它们赋值为 T. 这样, 每一个子句取值为真的概率至少为 $1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$. 若我们用 Z_j 这个随机变量表示第 j 个子句是否被满足 (满足取 1), 那么问题的解 $Z = \sum_j z_j$. 根据期望的线性性

$$\mathbb{E}\left[\sum_j z_j\right] = \sum_j \mathbb{E}[Z_j] = \frac{7}{8}k.$$

即算法在期望意义下能获得至少 $\frac{7}{8}k$ 个成真子句. ◇

注 3 以上算法有一个有意思的推论, 就是对任何一个 3-CNF, 都存在一个赋值使得其中 $\frac{7}{8}$ 比例的子句得到满足. 我们虽然知道这是真命题, 但是 (此时) 却无法具体构造出满足要求的赋值来. 这种和随机算法紧密相连的非构造性证明方法称为**概率方法** [Spe94]. ♤

注 4 不过, 我们确实可以设计一个期望运行时间为多项式的新算法, **保证**得到至少 $\frac{7}{8}k$ 个成真子句. 方法为不断重复上面的算法, 直到得到了 $\frac{7}{8}k$ 个成真子句为止.

分析: 根据几何分布的性质, 若上面随机算法成功的概率为 p, 则我们期望意义上需要运行 $1/p$ 次就能成功. 假设得到 j 个子句成真的概率为 p_j, 那么 $\frac{7}{8}k = \sum_j j p_j$. 用 k' 表

示最大的严格小于 $\frac{7}{8}k$ 的整数, 令 $p = \sum_{j\le k'} p_j$, 则

$$\frac{7}{8}k = \sum_j jp_j \le (1-p)k' + pk.$$

这里的不等号中, 我们把 $j \le k'$ 都放大到 k', 剩余的 j 都放大到 k, 于是

$$\frac{1}{p} \le \frac{k}{\frac{7}{8}k - k'} \le 8k,$$

即期望运行时间不超过 $8k$. ♤

下面这个例子顺便还用上了我们在引言中给出的求隐函数渐近式的方法.

例 5 系统中有 n 项工作需要 n 个处理器来处理, 我们希望每个处理器处理的工作尽量均匀. 如果有一个中心化的调度器的话, 那么问题是平凡的——只需要给每个处理器至多 1 项工作.

现在假设系统是分布式的, 就是没有一个中心化的调度器. 采用随机算法: 每个任务被等概率地分配给任何一个处理器. 显然算法无法保证分配一定是均匀的, 但是可以证明大概率是均匀的.

分析: 用 X_i 表示 i 得到的任务个数, 那么 $\mathbb{E}[X_i] = 1$. 注意 X_i 是 n 个 Bernoulli 变量的独立和, 所以由课本 Chernoff 界的第一条, 取 $\mu = 1, 1+\delta = c$, 则

$$\Pr[X_i > c] \le \frac{\mathrm{e}^{c-1}}{c^c}.$$

用 union bound, 所有处理器被分配到的任务个数 $\le c$ 的概率至少是

$$\Pr[X_1 \le c, \dots, X_n \le c] = 1 - \Pr\left[X_1 > c \vee \cdots \vee X_n > c\right] \ge 1 - \frac{n\mathrm{e}^{c-1}}{c^c}. \tag{1.1}$$

我们希望取适当的 c, 使得 $c^c \sim n$. 两边求两次对数得 $\log\log n = \log c + \log\log c \sim \log c$, 所以 $c\log c = \log n \sim c\log\log n$, 于是 $c = \Theta\left(\frac{\log n}{\log\log n}\right)$, 可以算出此时式 (1.1) 右侧是 $1 - O\left(\frac{1}{n}\right)$, 所以以高概率有: 每个处理器收到的任务不超过 $\Theta\left(\frac{\log n}{\log\log n}\right)$. ♢

2 随机算法的补充例题

注意: Las Vegas 和 Monte-Carlo 算法某种意义上是一样的. 一方面, 假设我们有一个错误概率 p 的运行时间不超过 $f(n)$ 的算法, 如果我们还能在 $g(n)$ 时间内检查解是否正确, 那么通过重复运行 + 检查便可得到 $(f(n)+g(n))/p$ 的 Las Vegas 算法 (下面第一小节经常用这种思想设计算法); 另一方面, 假设我们对问题有一个期望运行时间不超过 $f(n)$

的 Las Vegas 算法, 根据 Markov 不等式 Pr[运行时间 $\geq 3f(n)] \leq \frac{1}{3}$, 如果 $3f(n)$ 时间内它停不下来, 就强迫算法停止并乱猜一个答案, 否则就用算法的输出, 这样就得到错误概率至多 $\frac{1}{3}$ 的 Monte-Carlo 算法.

以下是几个比较简单的问题, 欲尝试更丰富的题目可参看 [MR95].

2.1 Las Vegas

问题 6 回忆图的 3-染色问题. 现在考虑这样一个改版: 用 3 种颜色给图中的顶点染色, 使得两个端点颜色不同的边尽可能多. 假设问题的最优解是 OPT. 请设计一个随机的近似算法, 使得期望意义下能获得至少 $\frac{2}{3}$OPT 的性能.

解 算法: 对每个顶点, 分别独立地以均匀的概率给顶点染色.

分析: 对任何一条边, 两个端点颜色不同的概率至少为 $1 - \frac{3}{3\times 3} = \frac{2}{3}$, 根据期望的线性性知算法期望意义上能保证 2/3 的边是异色的, 自然也有至少 $\frac{2}{3}$OPT 的性能. □

注 7 类似问题: 考虑在例 2 中不再要求 CNF 为 3-CNF, 即每个子句中可以不是 3 个文字. 给出 MAX-SAT 的一个随机算法, 使得期望意义下至少有 60% 的子句得到满足. (方法: 如果只有一个文字的子句之间有互相矛盾的 $((x_i) \wedge (\neg x_i))$, 就把其中的一个文字从所有的子句里去掉, 然后以 p 的概率给变元赋 T; 最后把去掉的文字适当地补回来; 计算出最优的 $p \approx 0.618$.) ♤

问题 8 给定 n 个正整数 $A = \{a_1, \ldots, a_n\}$ 和一组限制 $T \subseteq A \times A \times A$. 我们希望把 A 中元素适当地排成一列——限制 $(a_i, a_j, a_k) \in T$ 的意思是 a_j 最好能排在 a_i 和 a_k 的中间 (但对 a_i, a_k 的相对位置无要求). 请设计一个随机近似算法, 使得它能在多项式级别的期望运行时间内给出一个常数近似.

解 算法: 均匀随机地挑选 A 的一个排列, 检查它是否满足了至少 $\frac{1}{3}$ 比例的限制; 重复猜测直到真的满足了为止.

分析: 每个 $(a_i, a_j, a_k) \in T$ 被满足的概率是 $\frac{2}{3!} = \frac{1}{3}$, 所以期望意义下能满足 $\frac{1}{3}$. 设算法的成功概率为 p, 以及 ℓ 是最大的严格小于 $\frac{1}{3}|T|$ 的整数, 则仿照注 4 的方法可得

$$\frac{1}{3}|T| \leq \ell(1-p) + |T|p \leq \ell(1-p) + n^3 p \quad (\text{因为 } |T| \leq n^3),$$

移项根据 ℓ 的定义得出 $\frac{1}{3} \leq (n^3 - \ell)p$, 即 $1/p = O(n^3)$. □

注 9 类似问题: 给一个有 n 个顶点和 m 条边的图以及正整数 $k \leq n$, 求一个 k 顶点的导出子图, 使得该导出子图至少有 $\frac{m\binom{k}{2}}{\binom{n}{2}}$ 条边 (就是这个子图的稠密程度和整个图差不多). (随机选择 k 个点, 证明期望下边数符合要求, 再重复运行.) ♤

2.2 Monte-Carlo

问题 10 矩阵乘法验证: 给定矩阵 $A, B, C \in \mathbb{R}^{n\times n}$, 请给出一个运行时间 $O(n^2)$ 的 Monte-Carlo 算法, 验证是否有 $AB = C$.

解 算法: 每次都随机生成一个 n 维的布尔向量 $\boldsymbol{x}$ 并按括号顺序作乘法: $A(B\boldsymbol{x}), C\boldsymbol{x}$, 验证结果是否相等. 验证 k 次后如果都相等就宣布乘法正确, 否则宣布乘法错误.

执行 k 次只需要 $O(kn^2)$ 时间. 以下证明它宣布接受时, 出错的概率不超过 $\frac{1}{2^k}$. 实际上, 如果 $AB - C \neq \mathbf{0}$, 设其不为 $\mathbf{0}$ 的行有 $\boldsymbol{r}_{i_1}, \ldots, \boldsymbol{r}_{i_m}$, 由全概公式

$$\begin{aligned}
\Pr[x_{i_1}\boldsymbol{r}_{i_1} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} = \mathbf{0}] &= \Pr[x_{i_1}\boldsymbol{r}_{i_1} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} = \mathbf{0} \mid x_{i_2}\boldsymbol{r}_{i_2} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} = \mathbf{0}] \\
&\quad \times \Pr[x_{i_2}\boldsymbol{r}_{i_2} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} = \mathbf{0}] \\
&\quad + \Pr[x_{i_1}\boldsymbol{r}_{i_1} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} = \mathbf{0} \mid x_{i_2}\boldsymbol{r}_{i_2} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} \neq \mathbf{0}] \\
&\quad \times \Pr[x_{i_2}\boldsymbol{r}_{i_2} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} \neq \mathbf{0}] \\
&\quad \leq \frac{1}{2}(\Pr[x_{i_2}\boldsymbol{r}_{i_2} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} = \mathbf{0}) + \Pr[x_{i_2}\boldsymbol{r}_{i_2} + \cdots + x_{i_m}\boldsymbol{r}_{i_m} \neq \mathbf{0}]) \\
&\quad = \frac{1}{2}.
\end{aligned}$$

上式中的不等号是因为 $\boldsymbol{r}_{i_1} \neq \mathbf{0}$, 所以 x_{i_1} 的选择至多只有一个导致条件概率中的事件成立, 即条件概率均不超过 1/2. 根据上式, k 次执行后仍错误的概率不超过 2^{-k}. □

问题 11 近似中位数: 给定一个 n 元实数集合 S (n 充分大), 我们称 $x \in S$ 是 ε-近似的中位数, 如果分别至少有 $\left(\frac{1}{2} - \varepsilon\right) n$ 的 S 中元素大于和小于它. 考虑这样的随机算法: 从 S 中均匀随机抽出 c 个元素, 把这些元素 (集合 S') 的中位数作为原来集合的近似中位数. 证明: 给定 k, 可以适当地选择一个无关于 n 的 c, 算法以高概率得到 $\frac{1}{k}$-近似的中位数.

证明 为了计算得到 $\frac{1}{k}$-近似中位数的概率, 需要把 S' 中元素的分布和它们在 S 中的分布联系起来. 考虑把 S 中的元素从小到大分为 $Q_1, \ldots, Q_k$ 这些小段, 每段恰占据 $\frac{n}{k}$ 的比例. 我们考虑一种最理想的情况: 如果每个 $|S' \cap Q_i|$ 刚好在 $\frac{(1-2/k)c}{k}$ 和 $\frac{(1+2/k)c}{k}$ 之间, 那么 S' 的中位数就刚好落在 S 的中间 $\frac{2}{k}$ 一段, 即是 $\frac{1}{k}$-近似的中位数.

注意 $|S' \cap Q_i|$ 是 Bernoulli 变量的独立和, 所以利用 Chernoff 界 (课本第三式, 取 $\mu = c/k, C = 2/k$), 我们知道

$$\Pr\left[\left||S' \cap Q_i| - \frac{c}{k}\right| \geq \frac{c}{k}\frac{2}{k}\right] \leq 2\exp\left(-\min\left\{\frac{1}{k^2}, \frac{1}{k}\right\}\frac{c}{k}\right),$$

则由 union bound, 算法的失败概率不超过 $2k\exp(-c/k^3)$. 我们取 $c = k^3 \log^2 k$, 则算法以至少 $1 - O\left(\frac{1}{k}\right)$ 概率输出 $\frac{1}{k}$-近似的中位数. □

注 12 类似问题: Alice 和 Bob 在一个干扰的信道上传输秘密, 消息都是 $\Sigma = \{0, 1\}$ 的字符串. 假设密文为 $alpha = \alpha_1 \cdots \alpha_n \in \Sigma^*$, 密钥为一个函数 $\Sigma^* \times \Sigma \longrightarrow \Sigma$, 那么解密过程迭代地向前进行: $\beta_1 = f(\alpha_1), \beta_2 = f(\beta_1, \alpha_1), \ldots, \beta_n = f(\beta_1, \ldots, \beta_{n-1}, \alpha_n)$. 显然如果第 j 字符 β_j 传输错误, 则后面的解密都不能保证正确了. 假设传输密文的过程中, 每个字符都以 $\frac{1}{4}$ 的概率传错 (0 变成 1, 1 变成 0). 假设传输方可以多次重复传输 (每次用不同的密码, 但密文对应的明文都相同), 请设计一个算法, 使得只需要发送 $O(\log n)$ 次, 接收方就能以

$1 - O\left(\frac{1}{n}\right)$ 的概率得到完全正确的明文. (接收到的全部消息同时逐个字符解密, 每次得到一个新字符后, 这些解密结果中数量多的那个作为最终的明文对应的字符 (即 majority vote 算法), 第 i 位错误的概率相当于 Chernoff 界中的 $\Pr[X \geq 2\mu]$.) ♤

问题 13 (此题也可算作在线算法) 相亲问题: 张三去 BBS 上征友, 他一方面很希望得到 npy, 另一方面又怕 npy 不够好. 假设他**依次**遇到 n 个潜在的 npy, 遇到第 i 个同学时, 他就能为这个同学评估一个心仪程度 a_i. 但是他必须当场决定是否要和对方在一起. 如果接受, 征友结束, 之后的同学都不会来了; 如果拒绝, 虽然张三还能看后面的同学, 但当前这位同学就不会再理睬张三. 假设 n 位同学前来的顺序是均匀随机的, 请帮张三设计一个策略, 使得他以常数的概率和 n 个同学里最心仪的那个成为 npy.

解 方法 (可能是家喻户晓的): 先拒绝前面一半的同学, 记录下最高心仪程度; 如果后面遇到了比最高心仪程度还高的同学, 就赶紧在一起; 如果到最后一个同学都没有心仪程度更高的, 就沮丧而后悔地接受.

分析: 假设张三最喜欢的是西施, 其次喜欢的是东施, 那么如果东施出现在前一半中, 而西施出现在后一半中, 他就能和西施在一起. 而这个概率大致是

$$\frac{\frac{n}{2}\frac{n}{2}(n-2)!}{n!} \sim \frac{1}{4}.$$

(简单地说, 在线算法的设计方法论有二, 第一是如果看不到未来的输入, 就贪心; 第二是如果目前已知的输入能帮助推断未来的输入, 则需要利用.) □

参考文献

随机算法是一个庞大的、触角很长的领域, 对此可以参考 [MR95] 和 [Spe94]. 它在计算机系统和网络中有很多应用, 比如 caching、packet routing 等等, 详情请参看 [KT06] 中的其他例子.

[Kar93] D. R. Karger. “Global Min-Cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm”. 刊于: *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’93. Austin, Texas, USA: Society for Industrial and Applied Mathematics, 1993, pp. 21–30.

[KT06] J. Kleinberg and É. Tardos. *Algorithm Design*. Pearson Education, 2006. ISBN: 978-0-32-129535-4.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge International Series on Parallel Computation. Cambridge University Press, 1995. ISBN: 978-0-52-147465-8.

[MRT18] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. 2nd ed. MIT press, 2018. ISBN: 978-0-26-203940-6.

[Spe94] J. Spencer. *Ten Lectures on the Probabilistic Method*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1994. ISBN: 978-0-89-871325-1.

编写: WC
E-mail: wchang@pku.edu.cn