CS 598CSC: 大数据算法	讲座日期：2014年8月26日
讲师：Chandra Chekuri	记录员：Chandra Chekuri

1 引言/行政事务

- 课程网站： https://courses.engr.illinois.edu/cs598csc/fa2014/.
- 我们不会遵循特定的书籍。 请参考网站上的指引资源。
- 根据4-5个作业、记录一次讲座和课程项目进行评分。 细节待确定。

课程目标

大数据是当今的热门话题。 本课程的目标是学习一些在这个领域中有用的基本算法和分析技术。 其中一些技术是古老的，许多是在过去十五年左右发展起来的。我们希望涵盖一些主题：

- 流式处理、草图和抽样
- 维度降低
- 图的流处理
- 数值线性代数
- 压缩感知
- Map-Reduce模型和一些基本算法
- 属性测试
- 通过通信复杂性进行下界估计

上述主题中有太多的材料。 计划是简要介绍基础知识，以便激发进一步探索的动力。

2 流处理/一遍过程计算模型

在流处理模型中，我们假设输入数据以流的形式到达。 更正式地说，数据被假设为由 m个项目/标记/对象 $a_1, a_2, \ldots, a_m$组成。 一个简单的情况是每个标记都是在范围 $[1..n]$ 内的数字。 另一个例子是每个标记表示给定为 (u, v) 的图中的一条边，其中 u和 v是表示节点索引的整数。 标记按顺序一个接一个到达。 算法必须在看到下一个标记之前处理标记 x_i。 如果我们被允许存储所有的标记，那么我们就处于标准的计算模型中，我们可以访问整个输入。 然而，我们将假设算法可用的空间远小于 m个标记；通常是 m的次线性，或者在理想情况下是多对数(m)。 我们的目标是使用这种有限的空间来(近似)计算/估计各种感兴趣的数量。

令人惊讶的是，可以计算出各种有用的函数。 流式算法的一些感兴趣的参数是：

- 算法使用的空间与流长度 m和令牌的性质有关
- 处理每个令牌的最坏情况时间
- 处理所有令牌的总时间（等效地处理一个令牌的摊销时间）
- 输出的准确性
- 成功获得所需近似的概率（假设算法是随机的）

有几个应用领域促使了流式模型。 在网络中，大量的数据包经过交换机，我们可能想要分析一些数据包的统计信息。
经过交换机的数据包数量远远超过了可以存储的离线处理数据包的数量。 在大型数据库中，数据存储在磁盘上，随机访问是不可行的；主内存的大小远小于磁盘上可用的存储空间。 一次通过或多次通过的流式算法可以避免在磁盘上使用复杂的数据结构。 还有一些应用程序中，数据分布在多个位置，我们需要分别处理它们，并以低通信开销合并结果。

在数据库应用程序和类似的领域中，讨论多次遍历模型或半流模型也是有意义的。 通常我们会假设遍历次数是一个小常数。

3 概率和不等式的背景知识

本课程将大量依赖概率方法。 我们将主要依赖离散概率空间。 我们将尽可能以高层次的方式进行讨论，并以黑盒的方式使用某些结果。

设 Ω 是一个有限集合。概率测度 p为每个$\omega \in \Omega$ 分配一个非负数 $p(\omega)$，使得 $\sum$ 元组 $(_\Omega, p)$ 定义了一个离散概率空间；该空间中的事件是 $A \subseteq_\Omega$的任意子集，事件的概率简单地为 $p(A) = \sum$ 当Ω是一个连续空间，例如区间[0,1]时，情况会变得更加复杂，我们需要讨论关于Ω上的测度空间 σ-代数；我们只能对Ω的某些子集分配概率。 我们不会深入细节，因为我们在这门课程中不需要任何形式的机器。

一个重要的定义是随机变量的定义。 在这门课程中，我们只关注实值随机变量。 在概率空间中，随机变量 X是一个函数 $X: \Omega \to \mathbb{R}$。在离散情况下，随机变量 X的期望值，表示为 $\mathbf{E}[X]$，被定义为 $\sum$ 对于连续空间 $\mathbf{E}[X] = \int X(\omega)dp(\omega)$ 与积分的适当定义。X的方差，用$\mathbf{V}\mathrm{ar}[X]$或σX^2表示，定义为$\mathbf{E}[(X-\mathbf{E}[X])^2]$。 标准差是$\sigma X$，方差的平方根。

定理1（马尔可夫不等式）假设X是一个非负随机变量，使得$\mathbf{E}[X]$是有限的。 那么对于任意的$t > 0$，$\mathrm{P}\mathbf{r}[X \geq t] \leq \mathbf{E}[X]/t$。

证明：证明在某种意义上是显然的，特别是在离散情况下。这里是一个概要。 定义一个新的随机变量Y，其中$Y(\omega) = X(\omega)$如果$X(\omega) < t$，并且$Y(\omega) = t$如果$X(\omega) \geq t$。Y是

非负且 $Y \leq X$逐点方式，因此 $\mathbf{E}[Y] \leq \mathbf{E}[X]$。我们还可以看到：

$$\begin{aligned}\mathbf{E}[X] \geq \mathbf{E}[Y] &= \sum_{\omega:X(\omega)<t} X(\omega)p(\omega) + \sum_{\omega:X(\omega)\geq t} tp(\omega) \\ &\geq t \sum_{\omega:X(\omega)\geq t} p(\omega) \qquad \text{（因为 } X\text{是非负的）} \\ &\geq t\Pr[X\geq t]。\end{aligned}$$

连续情况下，可以用积分替代求和。 □

马尔可夫不等式在该假设下是紧的。 假设你可以构造一个例子。 我们对随机变量了解得越多，就能更好地限制其偏离期望的程度。

定理**2**（切比雪夫不等式）设 X为一个具有期望值 $\mathbf{E}[X]$和方差 $\mathbf{Var}[X]$有限的随机变量。 那么 $\Pr[|X| \geq t] \leq \mathbf{E}[X^2]/t^2$，且 $\Pr[|X - \mathbf{E}[X]| \geq t\sigma_X] \leq 1/t^2$。

证明：考虑非负随机变量$Y = X^2$. $\Pr[|X| \geq t] = \Pr[Y \geq t^2]$，我们对后者应用马尔可夫不等式。通过考虑 $Y= (X - \mathbf{E}[X])^2$，可以得到类似的第二个不等式。

□

Chernoff-Hoeffding不等式：我们将多次使用适用于由有界和独立随机变量组成的随机变量的**Chernoff-Hoeffding**不等式。 这些不等式有几个版本。 首先，我们陈述一个适用于非负随机变量且与变量数量无关的一般不等式，它仅依赖于期望而不是变量数量。

定理**3**（**Chernoff-Hoeffding**）：设 $X_1, X_2, \ldots,$ X_n为独立的二进制随机变量， $a_1, a_2, \ldots,$ a_n为 $[0, 1]$中的系数。设 $X = \sum_i a_i X_i$. 然后

- 对于任意的 $\mu \geq \mathbf{E}[X]$和任意的$\delta > 0$, $\Pr[X > (1+\delta)\mu] \left(\leq \frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^\mu$.
- 对于任意的 $\mu \leq \mathbf{E}[X]$和任意的$\delta > 0$, $\Pr[X < (1-\delta)\mu] \leq e^{-\mu\delta^2/2}$.

以下推论限制了偏离均值的范围。

推论 **4** 在定理 *3* 的条件下，以下成立*:*

- 如果$\delta > 2e - 1$, $\Pr[X \geq (1+\delta)\mu] \leq 2^{-(1+\delta)\mu}$.
- 对于任意的 U，存在一个常数 $c(U)$，使得对于 $0 < \delta < U$， $\Pr[X \geq (1+\delta)\mu] \leq e^{-c(U)\delta^2\mu}$.
  特别地，与下尾部界限相结合，

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-c(U)t^2\mu}.$$

我们将读者引用到随机算法的标准书籍[6]和[4]中，以得到上述界限的推导。

如果我们只对上尾部感兴趣，我们还有以下界限，它们显示了 μ对 n的依赖性，以获得一个反向多项式概率。

推论**5** 在定理*3*的条件下，存在一个通用常数 α，使得对于任意的$\mu \geq \max\{1, \mathbf{E}[X]\}$，以及足够大的 n和 $c \geq 1$， $\Pr[X > \frac{\alpha c \ln n}{\ln \ln n} \cdot \mu] \leq 1/n^c$. 同样地，存在一个常数 α使得对于任意的 $\epsilon > 0$， $\Pr[X \geq (1+\epsilon)\mu + \alpha c \log n/\epsilon] \leq 1/n^c$.

备注 **6** 如果 X_i在范围 $[0, b]$内，其中 b不等于 1，可以适当地缩放它们并使用标准界限。

有时候我们需要处理在范围 $[-1,1]$内的随机变量。 考虑 $X = \sum_i X_i$，其中对于每个 i， $X_i \in [-1,1]$且 $\mathbf{E}[X_i] = 0$，且 X_i是独立的。在这种情况下 $\mathbf{E}[X] = 0$，我们不能再期望一个无维度的界限。 假设每个 X_i以概率$1/2$为1，以概率$1/2$为-1。那么 $X = \sum_i X_i$对应于一个一维的随机游走，尽管期望值为0，但 X的标准差是$\Theta(\sqrt{n})$。 可以证明 $\Pr[|X| \geq t\sqrt{n}] \leq 2e^{-t^2/2}$。 对于这些设置，我们可以使用以下界限。

定理**7**令 $X_1, X_2, \ldots, X_n$是独立的随机变量，对于每个 i， $X_i \in [a_i, b_i]$。令 $X = \sum_i a_i X_i$并且让 $\mu = \mathbf{E}[X]$。然后

$$\Pr[|X - \mu| \geq t] \leq 2e^{-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}}.$$

特别是如果 $b_i - a_i \leq 1$对于所有 i的话

$$\Pr[|X - \mu| \geq t] \leq 2e^{-\frac{2t^2}{n}}$$

注意 $\mathbf{Var}[X] = \sum_i$可以证明一个基于以下形式的界

$$\Pr[|X - \mu| \geq t] \leq 2e^{-\frac{t^2}{2(\sigma_X^2 + Mt/3)}}$$

其中 $|X_i| \leq M$对于所有 i。

备注**8**将*Chebyshev*界限与具有相同方差的*Chernoff-Hoeffding*界限进行比较。

统计估计器，降低方差和提升：在流计算模型中，我们主要使用随机算法来计算数据流中的函数 f of the data stream $x_1, \ldots, x_m$. 它们通常通过产生一个无偏估计器，通过随机变量 X来计算函数值。 也就是说，算法具有 $\mathbf{E}[X]$是所需值的属性。 请注意，随机性是算法内部的一部分，而不是输入的一部分（我们稍后还将讨论在考虑随机顺序流时的输入随机性）。 拥有一个估计器通常是没有用的。 我们通常还会尝试评估 $\mathbf{Var}[X]$，然后我们可以使用切比雪夫不等式。

减少估计的方差的一种方法是并行运行算法（使用单独的随机位）并获得估计器 $X_1, X_2, \ldots, X_h$并使用 $X = \frac{1}{h} \sum_i X_i$ 作为最终估计器。 注意 $\mathbf{Var}(X) = \frac{1}{h} \sum_i \mathbf{Var}(X_i)$ 因为 X_i是独立的。 因此方差减小了一个因子 h。 另一种方法是使用 $X_1, X_2, \ldots$ 的中位数值， X_h作为最终估计器。 然后我们可以使用Chernoff-Hoeffding不等式来获得对 h的更好依赖性。实际上，这两种方法可以结合起来，在下一节中我们通过一个具体的例子来说明。

4 概率计数和Morris算法

假设我们有一个我们想要计数的长序列的事件。 如果我们想要计数一个长度最多为 N的事件序列，我们可以使用一个具有 $\lceil \log_2 N \rceil$位的计数器轻松实现。 在一些感兴趣的环境中，我们希望进一步减少这个计数。 不难证明，如果我们想要一个精确和确定性的计数，那么我们确实需要一个具有 $\lceil \log_2 N \rceil$位的计数器。 令人惊讶的是，如果我们允许近似和随机化，我们可以用大约$\log logN$位进行计数。 这最初是在Morris的一篇简短而精彩的论文中展示的[5]；这是一篇阅读历史动机的好论文。

Morris的算法保持一个计数器，基本上估计了$\log N$ 其中 N 是事件的数量，这需要大约$\log logN$位。 有几种变体，这里我们将讨论简单的一种，并指向[5, 3, 1]以获取其他方案和更详细和仔细的分析。

概率计数：
$X \leftarrow 0$
当（有新事件到达时）
  抛一枚有偏置的硬币，正面的概率是$1/2^X$
  如果（硬币正面朝上）
    $X \leftarrow X + 1$
结束当
将$2^X - 1$作为流的长度的估计输出。

对于整数 $i \geq 0$，让 X_n是随机变量，表示在 $i$events之后计数器的值。让 $Y_n = 2^{X_n}$。 下面的引理表明算法的输出是我们所需计数的无偏估计。

引理 9 $\mathbf{E}[Y_n] = n + 1.$

证明:对 n进行归纳。 当 $n = 0, 1$ 时容易验证，因为在这两种情况下，我们都有 Y_n确定地等于 $n+1$。对于 $n \geq 2$，我们有:

$$\begin{aligned}
\mathbf{E}[Y_n] = \mathbf{E}[2^{X_n}] &= \sum_{j=0}^{\infty} 2^j \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} 2^j \left(\Pr[X_{n-1} = j] \cdot (1 - \frac{1}{2^j}) + \Pr[X_{n-1} = j-1] \cdot \frac{1}{2^{j-1}} \right) \\
&= \sum_{j=0}^{\infty} 2^j \Pr[X_{n-1} = j] + \sum_{j=0}^{\infty} (2 \Pr[X_{n-1} = j-1] - \Pr[X_{n-1} = j]) \\
&= \mathbf{E}[Y_{n-1}] + 1 \\
&= n + 1
\end{aligned}$$

我们使用归纳法得到最终的等式。 □

由于 $\mathbf{E}[Y_n] = n+1$，我们有 $\mathbf{E}[X_n] = \log_2(n+1)$，这意味着在 n个事件之后，计数器中的位数的期望是$\log \log n + O(1)$位。 我们还应计算Y_n的方差。

引理10 $\mathbf{E}[Y_n^2] = \frac{3}{2} n^2 + \frac{3}{2} n + 1$ *and* $\mathbf{Var}[Y_n] = n(n-1)/2$。

证明：证明通过对 n的归纳进行。 容易验证基本情况 n= 0,1，因为 $Y_n = n+1$是确定的。对于 $n \geq 2$：

$$\begin{aligned}
\mathbf{E}[Y_n^2] = \mathbf{E}[2^{2X_n}] &= \sum_{j\geq 0} 2^{2j} \cdot \Pr[X_n = j] \\
&= \sum_{j\geq 0} 2^{2j} \cdot \left(\Pr[X_{n-1} = j](1 - \frac{1}{2^j}) + \Pr[X_{n-1} = j-1]\frac{1}{2^{j-1}} \right) \\
&= \sum_{j\geq 0} 2^{2j} \cdot \Pr[X_{n-1} = j] + \sum_{j\geq 0} \left(-2^j \Pr[X_{n-1} = j-1] + 4 2^{j-1} \Pr[X_{n-1} = j-1] \right) \\
&= \mathbf{E}[Y_{n-1}^2] + 3\mathbf{E}[Y_{n-1}] \\
&= \frac{3}{2}(n-1)^2 + \frac{3}{2}(n-1) + 1 + 3n \\
&= \frac{3}{2}n^2 + \frac{3}{2}\text{加}1.
\end{aligned}$$

我们使用归纳法和之前计算的 $\mathbf{E}[Y_n]$的值。$\mathbf{V}\mathrm{ar}[Y_n] = \mathbf{E}[Y_n^2] - (\mathbf{E}[Y_n])^2$
并且可以验证它等于 $n(n-1)/2$. □

4.1 近似和成功概率

期望值的分析表明算法的输出是一个无偏估计量。
方差的分析表明估计量是相当合理的。 然而，我们希望有一个更细致的理解。 例如，给定某个参数$\epsilon > 0$，$\Pr[|Y_n - (n+1)| > \epsilon n]$是多少？ 我们还可以提出一个相关的问题。 是否存在一个算法，给定□，δ保证输出 Y满足属性$\Pr[|Y - n| > \epsilon n] \leq \delta$，同时确保计数器的大小为 $O(\log \log n)$；当然，我们期望 $O()$符号中的常数取决于□，δ。

该算法可以通过使用相关方案来修改，以获得$(1+\epsilon)$的近似值，并以恒定的概率递增计数器
对于某个参数$_{aX}$，可以参考[5, ?]。 为了实现$(1+\epsilon)$的近似值，所需的计数器位数的期望可以证明为$\log \log n + O(\log 1/\epsilon)$位。

在这里，我们描述了两种通用方法，通过使用独立估计器来获得更好的近似值。 假设我们以独立的随机性并行运行基本算法 h次，以获得估计器 $Y_{n,1}, Y_{n,2}, \ldots, Y_{n,h}$。 然后我们输出 $Z = \frac{1}{h}$
$_{i=1} Y_{n,i} - 1$作为我们对 n的估计。 注意$\sum$ Z也是一个无偏估计器。 我们有 $\mathbf{Var}[Z] = \frac{1}{h}\mathbf{Var}[Y_n]$。

声明 **11** 假设 $h = 2/\epsilon^2$那么 $\Pr[|Z - n| \geq \epsilon n] < {}^1_4$.

证明:我们应用切比雪夫不等式得到

$$\Pr[|Z - n| \geq \epsilon n] \leq \frac{\mathbf{Var}[Z]}{\epsilon^2 n^2} \leq \frac{1}{h}\frac{\mathbf{Var}[Y_n]}{\epsilon^2 n^2} \leq \frac{1}{h}\frac{n(n-1)}{2\epsilon^2 n^2} < \frac{1}{4}.$$

□

现在假设我们想要对近似结果有高概率的保证。 也就是说，我们希望估计器是一个$(1+\epsilon)$-近似，并且概率至少为$(1-\delta)$，对于一些给定的参数 δ。

我们选择 $\ell = c \log \frac{1}{\delta}$对于一些足够大的常数 c。 我们独立并行地获得估计器 Z_1，Z_2，...， Z_ℓ并输出估计器的中位数；让我们称 Z'为对应于中位数的随机变量。 我们将证明以下内容。

声明 **12** $\Pr[|Z' - n| \geq \epsilon n] \leq (1 - \delta)$.

证明：定义一个指示随机变量 $A_i, 1 \leq i \leq \ell$其中 A_i为1，如果 $|Z_i - n| \geq \epsilon n$. 根据声明 11，$\mathbf{Pr}[A_i = 1] < 1/4$. 令 $A = \sum_{i=1}^{\ell} A_i$因此 $\mathbf{E}[A] < \ell/4$. 我们还观察到只有当 $A \geq \ell/2$时，才有$|Z' - n| \geq \epsilon n$，也就是说，超过一半的 Z_i的偏差大于 ϵn. 我们可以应用Che**r**n**o**ff-H**o**effd$_i$ng不等式来上界$\mathbf{Pr}[A \geq \ell/2] = \Pr[A \geq (1 + \delta)\mu]$其中 $\delta = 1$和 $\mu = \ell/4 \geq \mathbf{E}[A]$. 根据定理3，这至多为$(e/4)^{\ell/4}$. 由于 $\ell = c \log \frac{1}{\delta}$，对于适当的 c选择，概率至多为 δ.

□

在并行运行估计时，总空间使用量为 $O(\frac{1}{\epsilon^2} \cdot \log \frac{1}{\delta} \cdot \log \log n)$.

5 蓄水池抽样

随机抽样是一种在各个领域中都非常有用的通用技术。 目标是从一组项目 N中选择一个小的子集 S，使得 S能够代表 N，并且通常情况下，随机抽样效果很好。 最简单的抽样过程是从大小为m的集合中均匀地选择大小为 k的样本，其中 $k \leq m$ (通常 $k \ll m$)。 我们可以考虑有放回和无放回的抽样。 如果整个数据集以随机访问的方式可用，这些都是标准且简单的过程 —— 在这里我们假设我们可以访问一个随机数生成器。

下面我们将描述一种简单而好用的技术，称为蓄水池抽样，从流中获取大小为 1 的均匀样本。

```
UniformSample:
s ← null
m ← 0
当 (流未结束)
  m ← m + 1
  x_m 是当前项
  抛一枚有偏倚的硬币，正面概率为1/m
  如果 (硬币正面朝上)
      s ← x_m
结束循环
输出  s 作为样本
```

引理**13** 令 m为流的长度。 算法的输出 s是均匀的。 也就是说，对于任意 $1 \leq j \leq m$， $\Pr[s = x_j] = 1/m$。

证明：我们观察到 $s = x_j$ 如果 x_j 在算法中被考虑到时被选择（这发生的概率为$1/j$），而且没有 $x_{j+1}, \ldots,$ x_m被选择来替换 x_j。 所有相关事件是独立的，我们可以计算：

$$\Pr[s = x_j] = 1/j \times \prod_{i>j} (1 - 1/i) = 1/m.$$

□

为了获得 k个带替换的样本，对于 $k = 1$的过程可以并行进行，具有独立的随机性。 现在我们考虑从流中获取 k个不重复的样本。
输出将存储在大小为 k的数组 S中。

无替换采样(k):

$S[1..k] \leftarrow$ null
$m \leftarrow 0$
　　当 (流未结束)
　$m \leftarrow m+1$
　x_m 是当前项目
　如果 $(m \le k)$
　　$S[m] \leftarrow x_m$
　否则
　　$r \leftarrow$在范围 $[1..m]$ 内的均匀随机数
　　如果 $(r \le k)$
　　$S[r] \leftarrow x_m$
endWhile
输出 S

另一种描述是，当项目 x_t到达时（对于$t > k$），我们决定以 k/t的概率选择它包含在 S中，并且如果被选择，则从 S中选择一个均匀元素来替换为 x_t。

练习：证明该算法从流中输出一个大小为 k的无重复随机样本。

加权抽样：现在我们考虑将问题推广到加权抽样。
假设流由 m个项目 $x_1, \ldots, x_m$组成，每个项目 j的权重 $w_j > 0$。 目标是按照权重比例选择 k个样本。 假设 $k=1$，则目标是选择一个项目s，使得$\Pr[s = x_j] = w_j/W$其中 $W = \sum_i w_i$。 将均匀抽样算法推广到这个问题并获得 k个带替换的样本也很容易。 更有趣的情况是当我们想要 k个无重复样本时。 这意味着的确切定义并不明显。下面是一个算法。 首先从 $x_1, \ldots, x_m$中按照权重比例获得一个均匀样本。 从集合中移除 s并在剩余集合中获得另一个均匀样本。重复k次以获得 k个项目的集合（假设$k < m$）。 被移除的 k个项目形成输出S。 现在我们想以流式方式获得一个按照相同分布的随机集合 S。

首先我们在下面描述一个随机离线算法。

无重复加权采样(k):

对于 i从1到 m的循环
　$r_i \leftarrow$在区间$(0,1)$内均匀随机生成的数
　$w'_i = r_i^{1/w_i}$
结束循环
按 w'_i值按降序对项目进行排序
输出排序后的前 k个项目

我们将保留迄今为止看到的最重 k个修改后的权重来在流模型中实现上述算法。现在进行分析。

为了获得一些直觉，我们提出以下主张。

主张**14** 让 r_1, r_2是在 $[0,1]$上均匀分布的独立随机变量，令 $X_1 = r_1^{1/w_1}$ 和 $X_2 = r_2^{1/w_2}$ 其中 w_1, $w_2 \geq 0$. 然后

$$\Pr[X_1 \leq X_2] = \frac{w_2}{w_1 + w_2}.$$

通过概率密度函数进行基本分析可以证明上述论断。 更具体地说，假设w >0. 考虑随机变量 $Y = r^{1/w}$，其中 r在(0,1)上均匀选择。 随机变量 Y的累积概率函数，

$$F_Y(t) = \Pr[Y \leq t] = \Pr[r^{1/w} \leq t] = \Pr[r \leq t^w] = t^w.$$

因此，密度函数 $f_Y(t)$ 是 wt^{w-1}. 因此

$$\Pr[X_1 \leq X_2] = \int_0^1 F_{Y_1}(t) f_{Y_2}(t) dt = \int_0^1 t^{w_1} w_2 t^{w_2 - 1} dt = \frac{w_2}{w_1 + w_2}.$$

我们现在提出一个更一般的陈述。

引理**15** 设 $r_1, r_2, \ldots, r_n$ 是独立的随机变量，每个都服从均匀分布在 $[0,1]$上。设 $X_i = r_i^{1/w_i}$ 对于 $1 \leq i \leq n$. 对于任意的 $\alpha \in [0,1]$

$$\Pr[X_1 \leq X_2 \ldots \leq X_n \leq \alpha] = \alpha^{w_1 + w_2 + \ldots + w_n} \cdot \prod_{i=1}^{n} \frac{w_i}{w_1 + \ldots + w_i}.$$

证明：对于 n的归纳。 对于 n= 1，$\mathbf{Pr}[X_1 \leq \alpha] = F_Y$ $_1$假设引理对于所有的h $< n$成立我们证明对于n也成立.

$$\begin{aligned}
\Pr[X_1 \leq \ldots \leq X_n \leq \alpha] &= \int_0^\alpha \Pr[X_1 \leq \ldots \leq X_{n-1} \leq t] f_{Y_n}(t) dt \\
&= \int_0^\alpha t^{w_1 + w_2 + \ldots + w_{n-1}} \cdot \left(\prod_{i=1}^{n-1} \frac{w_i}{w_1 + \ldots + w_i} \right) w_n t^{w_n - 1} dt \\
&= w_n \left(\prod_{i=1}^{n-1} \frac{w_i}{w_1 + \ldots + w_i} \right) \int_0^\alpha t^{w_1 + w_2 + \ldots + w_n - 1} dt \\
&= \alpha^{w_1 + w_2 + \ldots + w_n} \cdot \prod_{i=1}^{n} \frac{w_i}{w_1 + \ldots + w_i}.
\end{aligned}$$

我们在第二个等式中使用了归纳假设. □

现在我们准备完成证明. 考虑任意固定的 j. 我们声称 X_j 是 $X_1, X_2, \ldots, X_m$ 中最大的数的概率等于 $\frac{w_j}{w_1 + \ldots + w_n}$. 你明白为什么吗? 在条件$X_j$是最大的情况下，其他变量仍然是独立的，我们可以再次应用这个观察。 你应该希望相信，在值X_1，X_2，...中选择最大的 k时 ，X_m会得到所需的样本。

参考文献注释：M**orris**的算法来自[5]。 详细分析请参见[3]，更清晰更简单的最新处理请参见[1]。
加权蓄水池抽样来自[2]。
更多关于高效蓄水池抽样方法请参见[7]。

参考文献

[1] Mikl´os Cs¨ur¨os. 使用浮点计数器进行近似计数。 Co*RR*, abs/0904.3062, 2009.

[2] Pavlos S Efraimidis和Paul G Spirakis. 带有蓄水池的加权随机抽样。 *I*nfor-*mation Processing Letters*, 97(5):181–185, 2006.

[3] Philippe Flajolet. 近似计数：详细分析。 *BIT* 数值数学，25(1)：113–134，1985年。

[4] Michael Mitzenmacher 和 Eli Upfal. 概率与计算：随机算法和概率分析。 剑桥大学出版社，2005年。

[5] Robert Morris. 在小寄存器中计数大量事件。 *A*CM 通信，21(10)：840–842，1978年。

[6] Rajeev Motwani 和 Prabhakar Raghavan. 随机算法。 剑桥大学出版社，1995年。

[7] Jeffrey S Vitter. 使用水库的随机抽样。 *ACM* 数学软件交易，11(1)：37–57，1985年。

CS 598CSC: 大数据算法	讲座日期：2014年8月28日
讲师：Chandra Chekuri	记录员：Chandra Chekuri

1 在流中估计频率矩

流计算文献中的一个重要部分是关于估计频率矩的问题。
令 $\sigma = a_1, a_2, \ldots, a_m$是一个数字流，其中对于每个 i， a_i是介于1和 m之间的整数。我们将尽量使用m表示流的长度， n表示整数范围为1。 令 f_i是流中整数 i的出现次数（或频率）。 我们定义f $= (f^1, f_2, \ldots, f\ n)$ 为给定流 σ的频率向量。 对于 $k \geq 0$， $F_k(\sigma)$ 被定义为 σ的第 k个频率矩：$F_k =$

$$\sum_i f_i^k.$$

我们将讨论几种算法来估计 F_k对于不同的 k值。 例如 F_0就是简单地计算 σ中不同元素的数量。注意 $F_1 = \sum_i f_i = m$，即流的长度。 当 k增大到 ∞时， F_k将集中在最频繁的元素上，我们可以将 F_∞看作是找到最频繁元素的过程。

定义1让 $\mathcal{A})(\sigma)$是随机流算法在流σ上的实数输出。如果对于实数函数 g， $\mathcal{A}$提供了一个 (α, β)-近似，则我们说A提供了(α, β)-近似。

$$\Pr\left[|\frac{\mathcal{A}(\sigma)}{g(\sigma)} - 1| > \alpha\right] \leq \beta$$

对于所有的 σ。

我们的理想目标是获得一个对于任意给定的$\epsilon, \delta \in (0, 1)$的$(\epsilon, \delta)$-近似。

2 哈希的背景

哈希技术在流式计算中起着基础性的作用，特别是用于估计频率矩。 我们将从理论角度简要回顾哈希，并特别关注 k-通用哈希。

哈希函数将有限的宇宙 $\mathcal{U}$映射到某个范围 $\mathcal{R}$。 通常范围是哈希表中的桶的集合，例如整数范围$[0..L-1]$（这里 L是哈希表中的桶数）。 为了方便理解，有时候我们会考虑将 $\mathcal{U}$映射到连续区间[0,1]的哈希函数。 一般情况下，我们将使用哈希函数族 $\mathcal{H}$，并且从 $\mathcal{H}$中均匀随机选择哈希函数 h；算法的分析将基于 $\mathcal{H}$的属性。 我们希望 $\mathcal{H}$具有两个重要且矛盾的属性：

- 从 $\mathcal{H}$中随机选择的函数应该像从 $\mathcal{U}$到范围的完全随机函数一样行为。
- $\mathcal{H}$应该具有良好的计算性质：

1许多流式计算论文中翻转了符号的使用，使用n表示流的长度，使用m表示流中整数的范围

–从$\mathcal{H}$中均匀随机选择一个函数应该很容易

–任何函数 $h \in \mathcal{H}$ 的表示大小都应该很小，以便可以紧凑存储

–评估函数 h应该是高效的

定义**2**一组随机变量 $X_1, X_2, \ldots, X_n$是 k-次独立的，如果变量 X_i $_1, X_{i_2}, \ldots, X_{i_k}$ 对于任何不同的指标集合 $i_1, i_2, \ldots, i_k$，它们是独立的。

取三个随机变量 $\{X_1, X_2, X_3\}$其中 X_1, X_2是独立的 $\{0, 1\}$随机变量，且 $X_3 = X_1 \oplus X_2$。 很容易检查这三个变量是两两独立的，尽管它们不是完全独立的。

在Carter和Wegman [3]的工作之后，k-通用哈希族的类别，特别是对于k= 2，提供了一个很好的权衡。 如果从$\mathcal{H}$中随机选择一个函数h，则$\mathcal{H}$是强2-通用的，如果满足以下属性：（i）对于每个$x \in \mathcal{U}$，$h(x)$（这是一个随机变量）在范围内均匀分布，并且（ii）对于每对不同的$x, y \in \mathcal{U}$，$h(x)$和$h(y)$是独立的。2-通用哈希族也称为成对独立哈希族。. 弱2-通用族满足$\Pr[h(x) = h(y)] = 1/L$对于任何不同的x, y。 我们对成对独立随机变量做出了一个重要观察。

引理 **1** 令 $Y = \sum_{i=1}^{h} X_i$其中 $X_1, X_2, \ldots, X_h$是两两独立的。那么

$$\mathbf{Var}[Y] = \sum_{i=1}^{h} \mathbf{Var}[X_i].$$

此外，如果 X_i是二进制/指示随机变量，则

$$\mathbf{Var}[Y] \leq \sum_i \mathbf{E}[X_i^2] = \sum_i \mathbf{E}[X_i] = \mathbf{E}[Y].$$

有一个简单而美妙的构造方法可以生成两两独立的哈希函数。 令 p是一个大于等于 $|\mathcal{U}|$ 的质数。 回顾一下， $Z\, p = \{0, 1, \ldots, p-1\}$在标准的模加法和模乘法下构成一个域。 对于每个$a, b \in [_p]$，我们可以定义一个哈希函数 $h_{a,b}$其中$h_{a,b}(x) = ax + b \mod p$。 我们可以看到，我们只需要存储两个$\Theta(\log p)$位的数字$a, b$来隐式存储$h_{a,b}$，并且对 $h_{a,b}(x)$的计算只需要一次加法和一次乘法的$\log\, p$位数。 此外，从 $\mathcal{H}$中随机选择一个哈希函数需要随机选择a, b，这也很容易。 我们声称 $\mathcal{H}$是一个两两独立的族。 你可以通过观察得出这一点，对于不同的x, y和任意的i, j这两个方程 $ax + b = i$和 $ay + b = j$同时有一个唯一的a, b解。 注意，如果 $a = 0$，那么这个哈希函数就没有什么用处；所有的元素都被映射到 b上。 然而，为了使 $\mathcal{H}$成为两两独立的，需要包括这些哈希函数，但是 $a = 0$ 的概率是 $1/p$，而 $\mathcal{H}$中有p^2个函数。 如果只想要一个弱一致性哈希族，我们可以从 $[1..(\, p - 1)]$ 中选择 a。 哈希函数的范围是 $[\, p]$。 为了将范围限制在 L内，我们可以定义 $h'_{a,b}(x) = (ax + b \mod p) \bmod L$。更一般地说，我们将说$\mathcal{H}$是 k-一致的，如果每个元素在范围内均匀分布，并且对于任意的 k个元素 $x_1, \ldots, x_k$，随机变量 $h(x_1), \ldots, h(x_k)$ 是独立的。

假设 $\mathcal{U}$是整数集合$[0..|\mathcal{U}|]$，对于任意固定 k，存在 k-通用哈希族的构造，使得族中的每个哈希函数 h可以使用 $O(k \log |\mathcal{U}|)$位（基本上是 k个数字）存储，并且可以使用 $O(k)$个算术操作在$\log |\mathcal{U}|$位数字上进行评估。 我们将忽略具体的实现细节，并将读者引用到关于哈希的大量文献以获取更多细节。

3 估计不同元素的数量

精确计数确定性算法的下界：我们认为任何精确计算不同元素数量的确定性流算法需要 $\Omega(n)$ 位。为了看到这一点，假设存在一个使用严格少于 n位的算法 $\mathcal{A}$。 考虑 $h= 2^n$个不同的流 σ_S，其中 $S \subseteq [n]$； σ_S由 S的元素以任意顺序组成。 由于$\mathcal{A}$使用了 n-1位或更少，必然存在两个不同的集合 S_1，S_2，使得 σ_S的结束时 $\mathcal{A}$的状态相同。

$\sigma_{S_1}, \sigma_{S_2}$ 是相同的。由于 S_1, S_2是不同的，所以存在一个元素 i在 $S_1 \setminus S_2$或 $S_2 \setminus S_1$；不失一般性，假设是前者。 那么很容易看出 $\mathcal{A}$不能正确计算两个流中至少一个的数量，$< \sigma_S$ $_1, i >, < \sigma_{S_2}, i >$.

基本的哈希思想：我们现在讨论一种简单的高层次思想来估计流中不同元素的数量。 假设 h是一个理想的随机哈希函数，将[1..n]映射到区间[0,1]。 假设流中有 d个不同的元素 $\sigma = a_1, a_2, \ldots, a_m$。 如果 h的行为类似于随机函数，那么集合 $\{h(a_1), \ldots, h(a_m)\}$将表现为在[0,1]上独立均匀分布的 d个独立随机变量的集合。 令 $\theta= \min\{h(a_1), \ldots, h(a_m)\}$； θ的期望值为

$\frac{1}{d+1}$因此 $1/\theta$是一个好的估计量。 在流计算模型中，我们可以通过对每个传入的值进行哈希并跟踪最小值来计算 θ。 我们只需要在内存中保存一个数字。 虽然简单，但该算法假设理想的哈希函数，而我们只有一个无偏估计量。 要将这个想法转化为一个具有适当保证的可实现算法，需要进行一些工作。 关于这个问题有几篇论文，我们现在将讨论其中一些方法。

3.1 AMS 算法

在这里，我们描述了一个具有更好参数的算法，但它只提供一个常数因子的近似解。 这归功于 Alon、Matias 和 Szegedy 在他们关于估计频率矩的著名论文[1]中。我们需要一些符号表示。 对于一个整数$t >0$，让 zeros(t) 表示二进制表示的 t以零结尾的零的个数；等价地

$$\text{zeros}(t) = \max\{i :| \ 2^i\text{可以整除 } t\}。$$

AMS-不同元素:

$\mathcal{H}$是一个从 $[n]$ 到 $[n]$ 的2-全域哈希族
从 $\mathcal{H}$ 中随机选择一个 h
$z \leftarrow 0$
当 (流不为空) 执行
  a_i 是当前项
  $z \leftarrow \max\{z, \text{zeros}(h(a_i))\}$
结束
输出 2 $^{z+\frac{1}{2}}$

首先，我们注意到每个元素的空间和时间复杂度为 $O(\log n)$. 现在我们来分析输出的近似质量. 回想一下， $h(a_j)$ 在 $[n]$ 上均匀分布. 为了简单起见，我们假设 n是2的幂.

令 d表示流中不同元素的数量，并将它们表示为 $b_1, b_2, \ldots, b_d$. 对于给定的 r，令 $X_{r,j}$为指示随机变量，如果 $\text{zeros}(h(b_j)) \geq r$，则为1. 令 $Y_r = \sum$ $_j X_{r,j}$. 也就是说， Y_r是具有至少 r个零的不同元素的数量。

由于 $h(b_j)$ 在 $[n]$ 中均匀分布，

$$\mathbf{E}[X_{r,j}] = \Pr[\text{zeros}(h(b_j)) \geq r] = \frac{(n/2^r)}{n} \geq \frac{1}{2^r}.$$

因此

$$\mathbf{E}[Y_r] = \sum_j \mathbf{E}[X_{r,j}] = \frac{d}{2^r}.$$

因此我们有 $\mathbf{E}[Y_{\log d}] = 1$ (假设 d 是2的幂次方)。
现在我们计算 Y_r的方差。 注意，变量 $X_{r,j}$和 $X_{r,j'}$是两两独立的，因为 $\mathcal{H}$是2-通用的。因此

$$\mathbf{Var}[Y_r] = \sum_j \mathbf{Var}[X_{r,j}] \leq \sum_j \mathbf{E}[X_{r,j}^2] = \sum_j \mathbf{E}[X_{r,j}] = \frac{d}{2^r}.$$

使用马尔可夫不等式

$$\Pr[Y_r > 0] = \Pr[Y_r \geq 1] \leq \mathbf{E}[Y_r] \leq \frac{d}{2^r}.$$

使用切比雪夫不等式

$$\Pr[Y_r = 0] = \Pr[|Y_r - \mathbf{E}[Y_r]| \geq \frac{d}{2^r}] \leq \frac{\mathbf{Var}[Y_r]}{(d/2^r)^2} \leq \frac{2^r}{d}.$$

让 z'在流的末尾的值，并让 $d'= 2\ z'^{+1}{}_2$为算法输出的估计值 d。 我们声称 d'与 d相比，在常数概率下不能太大。
让 a是最小的整数，使得$2^{a+1}{}_2 \geq 3d$。

$$\Pr[d' \geq 3d] = \Pr[Y_a > 0] \leq \frac{d}{2^a} \leq \frac{\sqrt{2}}{3}.$$

现在我们声称 d'与 d相比不会太小，具有恒定的概率。 为此，让 b是最大的整数，使得$2^{b+}{}_2{}^1 \leq d/3$。然后，

$$\Pr[d' \leq d/3] = \Pr[Y_{b+1} = 0] \leq \frac{2^{b+1}}{d} \leq \frac{\sqrt{2}}{3}.$$

因此，该算法提供了$(1/3, \sqrt{2}/3 \simeq 0.4714)$的近似值，用于不同元素的数量。使用中位数技巧，我们可以通过运行 $O(\log {}^1\delta)$个并行和独立的算法副本，使成功的概率至少为$(1 -_\delta)$，从而获得$(1/3, \delta)$的近似值。时间和空间复杂度将为 $O(\log \frac{1}{\delta} \log n)$。

3.2 A $(1-\epsilon)$-近似算法在 $O(\frac{1}{\epsilon^2} \log n)$ 空间

Bar-Yossef等人[2]描述了三种不同元素的算法，其中最后一种算法给出了一个界限 $\tilde{O}(\epsilon^2 + \log n))$空间和摊销时间每个元素 $O(\log n + \log \frac{1}{\epsilon})$; 符号 $\tilde{O})$抑制对$\log \log n$和$\log 1/\epsilon$的依赖。在这里，我们描述了他们的第一个算法，它给出了一个(ϵ, δ_0)-近似算法的空间 $O(\frac{1}{\epsilon^2} \log n)$和 $O(\log 1/\epsilon \log n)$每次更新的时间; 通过中位数技巧，再加上额外的$\log 1/\delta$因子，我们可以获得一个$(\epsilon, \delta)$-近似算法。

该算法基于Flajolet-Martin的哈希思想，将值映射到[0,1]并取最小值，但有一个小而重要的技术调整。 令 $t = {}_{\square}{}^{c}{}_2$，其中$c$是一个待定的常数。

BJKST-DistinctElements:

$\mathcal{H}$是一个从$[n]$到$[N = n^3]$的2-通用哈希族
从$\mathcal{H}$中随机选择一个h
$t \leftarrow \frac{c}{\epsilon^2}$
当 (流不为空) 时
  a_i 是当前项
  用 $h(a_i)$ 更新迄今为止见过的最小 $t$hash值
endWhile
令 v为哈希值中第 t小的值。
输出 tN/v。

我们观察到该算法可以通过跟踪 $t$hash值的方式实现，每个值占用 $O(\log n)$的空间，因此空间使用量为 $O(t \log n) = O(\frac{1}{\epsilon^2} \log n)$。 可以将 t值存储在二叉搜索树中，以便在处理新项时可以在 $O(\log t)$的时间内更新数据结构，总共需要 $O(\log \frac{1}{\epsilon} \log n)$的时间。

算法的直觉如下。 与之前一样，让 d是不同元素的数量并让它们是 $b_1, \ldots, b_d$。 哈希值 $h(b_1), \ldots, h(b_d)$ 在 [0,1] 上均匀分布。 对于任意固定的 t，我们预计大约有 t/d个哈希值落在区间 $[0, t/d]$ 中，而第 t小的哈希值 v应该大约是 t/d，这就证明了估计器对于 d的合理性是 t/v。 现在我们正式分析这个估计器的属性。

我们选择哈希族将 $[n]$ 映射到 $[n^3]$，因此至少以概率 $(1-1/n)$ ，随机哈希函数 h在 $[n]$ 上是单射的。 我们将假设这确实是情况。 此外，我们将假设 $n \geq \sqrt{\epsilon/24}$ 这意味着特别是 $\epsilon t/(4d) \geq 1/N$. 令 d'为算法返回的估计值
.

引理**2** $\Pr[d' < (1-\epsilon)d] \leq 1/6]$.

证明: 值 $h(b_1), \ldots, h(b_d)$ 在$1..N$中均匀分布 . 如果 $d' < (1-\epsilon)d$则意味着 $v > \frac{tN}{(1-\epsilon)d}$; 即小于 t个值落在区间$[1, \frac{tN}{(1-\epsilon)d}]$. 令 X_i为指示随机变量，表示事件 $h(b_i) \leq (1+\epsilon)tN/d$，令 $Y = \sum_{i=}^{d} X_i$.

由于 $h(b_i)$ 在 $[1..N]$ 上均匀分布，考虑到舍入误差，我们有 $(1+\epsilon/2)t/d \leq \mathbf{E}[X_i] \leq (1+3\epsilon/2)t/d$因此 $\mathbf{E}[Y] \geq t(1+\epsilon/2)$。 我们有 $\mathbf{Var}[Y] \leq \mathbf{E}[Y] \leq t(1+3\epsilon/2)$ (由于两两独立，引理1))。我们只有当$Y < t$且通过切比雪夫不等式，才有d′<(1−□)d。

$$\Pr[Y < t] \leq \Pr[|Y - \mathbf{E}[Y]| \geq \epsilon t/2] \leq \frac{4\mathbf{Var}[Y]}{\epsilon^2 t^2} \leq 12(\epsilon^2 t^2) \leq 1/6.$$

□

引理 **3** $\Pr[d' > (1+\epsilon)d] \leq 1/6]$.

证明: 假设 $d' > (1+\epsilon)d$, 即$v < \frac{tN}{(1+\epsilon)d}$ 这意味着超过 t个哈希值小于 $\frac{tN}{(1+\epsilon)d}$ 我们将证明这种情况发生的概率很小。

令 X_i为事件 $h(b_i) < (1-\epsilon/2)tN/d$的指示随机变量，令 $Y = \sum_{i=1}^{d} X_i$。我们有 $\mathbf{E}[X_i] < (1-\epsilon/2)t/d + 1/N \leq (1-\epsilon/4)t/d$（$1/N$是为了处理舍入误差）。正如我们所讨论的， $d' > (1+\epsilon)d$只有在$Y > t$时才会发生。根据切比雪夫不等式，

$$\Pr[Y > t] \leq \Pr[|Y - \mathbf{E}[Y]| \geq \epsilon t/4] \leq \frac{16\mathbf{Var}[Y]}{\epsilon^r t^2} \leq 16/(\epsilon^2 t^2) \leq 1/6.$$

□

参考文献注释：我们对AMS算法的描述来自Am**it**C**hak**r**abarti**的笔记。F**lajolet**和M**artin**在[5]中提出了基于哈希函数的基本算法。 [2] K**ane**, **Nelso**n **a**nd W**oodr**uff [7]对于固定的 δ_0描述了一个(ϵ, δ_0)-近似算法，空间复杂度为 $O(\frac{1}{\epsilon^2}+\log n)$(每次更新的时间也相同)。 这是一个理论上最优的算法，因为对于ϵ-近似，所需空间的下界为$\Omega(\ \epsilon^2)$和$\Omega(\mathbf{log}\, n)$。 我们将读者引用到[4, 6]中，以分析和实证评估一种被称为Hy**perLogLog**的算法，在实践中表现非常好。

参考文献

[1] Noga Alon, Yossi Matias, and Mario Szegedy. 近似频率矩阵的空间复杂度。计算机与系统科学杂志, 58(1):137–147, 1999. A*C*M ST*OC*1996的初步版本。

[2] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. 在数据流中计算不同元素的数量。 在计算机科学中的随机化和近似技术, 页1–10. S*p*ringer, 2002.

[3] J Lawrence Carter and Mark N Wegman. 哈希函数的通用类。 计算机与系统科学杂志, 18(2):143–154, 1979.. *A*CM *STO*C,1977的初步版本。

[4] Philippe Flajolet, Eric Fusy, Olivier Gandouet, Fr´ed´eric Meunier, et al. Hyperloglog: 一种近乎最优的基数估计算法的分析。算法分析*2007 (A*of*A07)*,页1*27*–146, *2007*.

[5] Philippe Flajolet 和 G Nigel Martin. 数据库应用的概率计数算法。计算机与系统科学杂志, 31(2):182–209, 1985.

[6] Stefan Heule, Marc Nunkesser 和 Alex Hall. Hyperloglog 在实践中: 一种最先进的基数估计算法的算法工程。 在*EDBT 2013* 会议中, 意大利热那亚, *2013*.

[7] Daniel M Kane, Jelani Nelson 和 David P Woodruff. 一种最优算法用于不同元素问题。 在第二十九届A*C*M *SIGMOD-SIGACT-SIGART*数据库系统原理研讨会, 页码 41–52. A*C*M, 2010.

CS 598CSC: 大数据算法	讲座日期: 2014年9月2日
讲师: Chandra Chekuri	记录员：Chandra Chekuri

1 AMS 抽样

我们已经看到了蓄水池抽样和相关的加权抽样技术，可以从流中获取独立的样本，而算法不知道流的长度。 现在我们讨论一种从流中抽样的技术 $\sigma = a_1, a_2, \ldots, a_m$其中 a_j是来自 $[n]$ 的整数，我们希望估计一个函数$g(\sigma) :=$

$$\sum_{i \in [n]} g(f_i)$$

其中 f_i是 i的频率， g是一个实值函数，满足 $g(0) = 0$. 一个自然的例子是估计频率矩 $F_k = \sum_{i \in [n]} f^k{}_i$；在这里，我们有 $g(x) = x^k$，对于 $k \geq 1$，这是一个凸函数。 另一个例子是 σ的经验熵，定义为 $\sum_{i \in [n]} p_i \log p_i$ where $p_i = \frac{f_i}{m}$ is the empirical probability of i; here $g(x) = x \log x$.[1]

AMS 从著名论文 [?] 中采样得到了 $g(\sigma)$ 的无偏估计。 该估计器基于一个随机变量 Y的定义如下。 令 J从 $[m]$ 中均匀随机抽样。
也就是说， R是在$_J$之后的标记数的计数，这些标记是相同坐标的。 然后，令 Y为以下估计值：

$$Y = m(g(R) - g(R-1))。$$

下面的引理表明 Y 是 $g(\sigma)$ 的无偏估计器。

引理 **1**

$$\mathbf{E}[Y] = g(\sigma) = \sum_{i \in [n]} g(f_i).$$

证明：当 $a_J = i$时， J是一个均匀采样，所以 $a_J =_i$的概率恰好是 $f\, i/m$。 此外，如果aJ=i 那么 R在 $[f_i]$ 上是一个均匀分布的随机变量。

$$\begin{aligned}
\mathbf{E}[Y] &= \sum_{i \in [n]} \Pr[a_J = i] \mathbf{E}[Y | a_J = i] \\
&= \sum_{i \in [n]} \frac{f_i}{m} \mathbf{E}[Y | a_J = i] \\
&= \sum_{i \in [n]} \frac{f_i}{m} \sum_{\ell=1}^{f_i} m \frac{1}{f_i} \left(g(\ell) - g(\ell - 1)\right) \\
&= \sum_{i \in [n]} g(f_i).
\end{aligned}$$

□

在流式计算模型中，可以使用蓄水池抽样的思想来使用小空间估计 Y的值，从而生成一个均匀样本。 下面给出了算法；每当选择一个新样本时，计数 R会被重置。

[1] 在熵的上下文中，按照惯例，当 $x = 0$时，

AMSEstimate:
$s \leftarrow$ null
$m \leftarrow 0$
$R \leftarrow 0$
当 (流未结束)
  $m \leftarrow m + 1$
  a_m 是当前项
  抛一枚有偏倚的硬币，正面概率为$1/m$
  如果 (硬币正面朝上)
    $s \leftarrow a_m$
    $R \leftarrow 1$
  否则如果 $(a_m == s)$
    $R \leftarrow R + 1$
结束循环
输出 $m(g(R) - g(R-1))$

为了通过估计器Y获得一个 (ϵ, δ)-近似值，我们需要估计 $\mathbf{Var}[\ Y]$ 并应用标准工具。 现在我们来处理频率矩时。

1.1 估计频率矩的应用

现在我们应用AMS采样来估计 F_k第 k个频率矩，其中 $k \geq 1$。 我们已经看到当我们设置 $g(x) = x_k$ 时， Y是 F^k的一个精确统计估计器。 现在我们在这个设置中估计 Y的方差。

引理2 当 $g(x) = x^k$ 且 $k \geq 1$,

$$\mathbf{Var}[Y] \leq kF_1F_{2k-1} \leq kn^{1-\frac{1}{k}}F_k^2.$$

证明:

$$\begin{aligned}
\mathbf{Var}[Y] &\leq \mathbf{E}[Y^2] \\
&\leq \sum_{i\in[n]} \Pr[a_J = i] \sum_{\ell=1}^{f_i} \frac{m^2}{f_i} \left(\ell^k - (\ell-1)^k\right)^2 \\
&\leq \sum_{i\in[n]} \frac{f_i}{m} \sum_{\ell=1}^{f_i} \frac{m^2}{f_i} (\ell^k - (\ell-1)^k)(\ell^k - (\ell-1)^k) \\
&\leq m \sum_{i\in[n]} \sum_{\ell=1}^{f_i} k\ell^{k-1}(\ell^k - (\ell-1)^k) \qquad (\text{使用 } (\ x^k - (x-1)^k) \leq kx^{k-1}) \\
&\leq km \sum_{i\in[n]} f_i^{k-1} f_i^k \\
&\leq kmF_{2k-1} = kF_1F_{2k-1}.
\end{aligned}$$

我们现在使用函数 x^k的凸性来证明第二部分。 注意 $\max_i f_i = F_\infty$.

$$F_1F_{2k-1} = (\sum_i f_i)(\sum_i f_i^{2k-1}) \leq (\sum_i f_i)F_\infty^{k-1}(\sum_i f_i^k) \leq (\sum_i f_i)(\sum_i f_i^k)^{\frac{k-1}{k}}(\sum_i f_i^k).$$

利用前面的不等式和不等式 $(\sum^n{}_{i=1} x_i)/n \leq ((\sum^n{}_{i=1} x^k{}_i)/n)^{\frac{1}{k}}$对于所有 $k \geq 1$（由于函数 $g(x) = x^k$的凸性），我们得到

$$F_1 F_{2k-1} \leq (\sum_i f_i)(\sum_i f_i^k)^{\frac{k-1}{k}}(\sum_i f_i^k) \leq n^{1-1/k}(\sum_i f_i^k)^{\frac{1}{k}}(\sum_i f_i^k)^{\frac{k-1}{k}}(\sum_i f_i^k) \leq n^{1-1/k}(\sum_i f_i^k)^2.$$

□

现在我们应用减小方差的技巧，然后再应用中位数技巧来得到一个高概率的界限。 如果我们对 Y取 h个独立的估计器，并取它们的平均值，方差会减小 h倍。我们令 $h = \frac{c}{\epsilon^2} k n^{1-1/k}$对于某个固定的常数 c。 令 Y'为得到的平均估计器。 我们有 $\mathbf{E}[Y'] = F_k$ 和 $\mathbf{Var}[Y'] \leq \mathbf{Var}[Y]/h \leq \frac{\epsilon^2}{c}$现在，使用切比雪夫，我们有

$$\Pr[|Y' - \mathbf{E}[Y']| \geq \epsilon \mathbf{E}[Y']] \leq \mathbf{Var}[Y']/(\epsilon^2 \mathbf{E}[Y']^2) \leq \frac{1}{c}.$$

我们可以选择 $c = 3$ 来获得一个 $(\epsilon, 1/3)$-近似解。 通过使用中位数技巧和 $\Theta(\log \frac{1}{\delta})$ 个独立的估计器，我们可以获得一个 (ϵ, δ)-近似解。 我们独立运行的估计器的总数是 $O(\log \frac{1}{\delta} \cdot \frac{1}{\epsilon^2} \cdot n^{1-1/k})$。 每个估计器需要 $O(\log n + \log m)$ 的空间，因为我们要跟踪 $[m]$ 中的一个索引，$[m]$ 中的一个计数，以及 $[n]$ 中的一个项目。 因此，获得一个 (ϵ, δ)-近似解的空间使用量是 $O(\log \frac{1}{\delta} \cdot \frac{1}{\epsilon^2} \cdot n^{1-1/k} \cdot (\log m + \log n))$。 处理每个流元素的时间也是相同的。

$\tilde{O}(n^{1-1/k})$的空间复杂度对于估计 F_k来说并不是最优的。 对于$k > 2$，可以达到$\tilde{O}(n^{1-2/k})$的最优解，实际上对于$1 \leq k \leq 2$，可以实现多对数空间。 我们将在课程后面看到这些结果。

参考文献：参见McGregor和Muthukrishnan的草稿书的第1章；参见AMS采样用于估计熵的应用。 详细解释频率矩的特殊情况，请参见Amit Chakrabarti的第5章。 特别是他提出了一个干净的引理，将方差减小技巧和中位数技巧捆绑在一起。

CS 598CSC: 大数据算法	讲座日期: 2014年9月2日
讲师: Chandra Chekuri	记录员：Chandra Chekuri

1 F_2 估计

我们已经看到了一个用于估计 F_k的通用算法，即流的 k'th频率矩，使用$\tilde{O}(n^{1-1/k})$-空间，对于 $k \geq 1$。 现在我们将看到一个令人惊讶地简单的算法，用于 F_2估计，由[2]提出。

AMS-F_2-估计:

$\mathcal{H}$是一个从$[n]$到 $\{-1, 1\}$的4-通用哈希族
从$\mathcal{H}$中随机选择h
$z \leftarrow 0$
当(stream不为空)时
  a_j 是当前项
  $z \leftarrow z + h(a_j)$
结束
输出 z^2

描述该算法的一个等效方式如下.

AMS-F_2-估计:

让 $Y_1, Y_2, \ldots, Y_n$是4-独立的随机变量
$z \leftarrow 0$
当(stream不为空)时
  a_j 是当前项
  $z \leftarrow z + Y_{a_j}$
结束
输出 z^2

两者之间的区别在于前者更适合流式处理. 不再保留$Y_1, \ldots, Y_n$我们从一个4-独立哈希族中随机选择h，以便可以紧凑地存储h在$O(\log n)$空间中，并且我们可以在$O(\log n)$时间内动态生成$Y_i = h(i)$。 我们将在第二种描述中分析该算法.

让 $Z = \sum_{i \in [n]} f_i Y_i$是表示流结束时 z值的随机变量。 请注意，对于所有 $i \in [n]$， $\mathbf{E}[Y_i] = 0$， $\mathbf{E}[Y_i^2] = 1$。此外，由于 $Y_1, \ldots, Y_n$是4-两两独立的，因此也是2-两两独立的，对于$_i =_{i'}$， $\mathbf{E}[Y\, iY\, i'] = 0$。 输出的期望值为

$$\mathbf{E}[Z^2] = \sum_{i,i' \in [n]} f_i f_{i'} \mathbf{E}[Y_i Y_{i'}] = \sum_{i \in [n]} f_i^2 \mathbf{E}[Y_i^2] + \sum_{i=i'} f_i f_{i'} \mathbf{E}[Y_i Y_{i'}] = \sum_{i \in [n]} f_i^2 = F_2.$$

我们还可以计算输出的方差，即 $\mathbf{E}[Z^4]$。

$$\mathbf{E}[Z^4] = \sum_{i \in [n]} \sum_{j \in [n]} \sum_{k \in [n]} \sum_{\ell \in [n]} f_i f_j f_k f_\ell \mathbf{E}[Y_i Y_j Y_k Y_\ell].$$

通过 Y的4次独立性，如果在多重集合中有一个索引在
i、j、k、ℓ中只出现一次，则E[YiYjYkYℓ] = 0，否则为1。 如果为1，则有两种情况：所有索引都相同，或者有两个不同的索引每个出现两次。因此，

$$\mathbf{E}[Z^4] = \sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]}\sum_{\ell\in[n]} f_i f_j f_k f_\ell \mathbf{E}[Y_i Y_j Y_k Y_\ell] = \sum_{i\in[n]} f_i^4 + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2 f_j^2.$$

因此，我们有

$$\begin{aligned}
\mathbf{Var}[Z^2] &= \mathbf{E}[Z^4] - (\mathbf{E}[Z^2])^2 \\
&= F_4 - F_2^2 + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2 f_j^2 \\
&= F_4 - (F_4 + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2 f_j^2) + 6\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2 f_j^2 \\
&= 4\sum_{i=1}^{n}\sum_{j=i+1}^{n} f_i^2 f_j^2 \\
&\leq 2F_2^2.
\end{aligned}$$

令 $X = Z^2$为输出估计。 我们现在应用平均化 $O(1/\epsilon_2)$估计的标准思想来减小方差，应用Cheb yshev不等式对平均估计器进行分析，以查看其具有 ϵ-近似度和 $> 1/\ 2$概率。 然后，我们应用中位数技巧，使用对数$(\frac{1}{\delta})$-独立的平均估计器来获得一个(ϵ, δ)-近似度。

整体空间需求为 $O(\ \frac{1}{\epsilon^2}\log\frac{1}{\delta}\log n)$，这也是处理每个元素的时间。

2 (线性) Sketching 和 Streaming with Updates

这个 F_2 估计算法非常简单，并具有以下有趣的特性。
假设 σ_1和 σ_2是两个流，算法在 σ_1和 σ_2上计算 z_1和 z_2。很容易看出，算法在 $\sigma = \sigma_1 \cdot \sigma_2$(两个流的连接)上计算 $z_1 + z_2$。
因此，算法将 z保留为流 σ的草图。 请注意，算法的输出不是 z，而是 z的某个函数(在 F^2 估计的情况下是 z_2)。 此外，在这种特殊情况下，草图是一个线性草图，我们稍后会更正式地定义。

形式上，流 σ的草图是一个数据结构 $z(\sigma)$，它具有以下属性：如果 $\sigma = \sigma \cdot \sigma_2$，则可以通过组合草图$z(\sigma_1)$和 $z(\sigma_2)$来计算 $z(\sigma)$。 理想情况下，组合算法应该占用很小的空间。 请注意，算法可以对草图进行后处理以输出估计值。

通过考虑比我们目前所见的更一般的流模型，可以说明草图算法的威力。我们已经考虑了形式为 a_1，a_2，...的流。 ，a_m其中每个a_i是一个令牌，特别是来自$[n]$的整数。 现在我们将考虑以下模型。 我们从一个 n维向量/信号 $\mathbf{x}$= $(0, 0, ..., 0)$开始，流令牌由对 $\mathbf{x}$的坐标的更新组成。 因此，每个令牌 a_t= (i_t, Δ_t)，其中 $i_t \in [n]$，而Δ_t是一个数字（可以是实数并且可以是负数）。 令牌 a_t更新了 $\mathbf{x}$的第 i个坐标：

$$x_{i_t} \leftarrow x_{i_t} + \Delta_t.$$

我们将让 $\mathbf{x}_t$成为 $\mathbf{x}$更新对应的值后的 $\mathbf{x}$ a的值，即 $\mathrm{a}_1, a_2, \ldots,$ a_t。如果 Δ_t允许
为负数，则该模型称为turnstile流；请注意 Δ_t为负数时允许删除项目。如果要求x$_t$始终为非负数，则我们有 $strict$turnstile流模型。进一步特殊情况是当 Δ_t要求为正数时，称为$cash\ register$模型。

线性草图特别简单但非常强大。线性草图对应于一个$k \times n$投影矩阵 M，向量 $\mathbf{x}$的草图简单地是 $M\mathbf{x}$。组合线性草图对应于简单地添加草图，因为 $M\mathbf{x}+M\mathbf{x}' = M(\mathbf{x}+\mathbf{x}')$。在流式设置中，当我们看到一个令牌 a_t= (i_t, Δ_t)时，更新草图相当于添加 $\Delta_t M\, e_i$
$_t$到草图
在这里 e_{i} 是向量，在第i行有1，在其他地方都是0。为了在小空间中实现算法，只需要能够高效地生成第i列的M，而不是存储整个矩阵M。

F_2估计作为线性草图：很容易看出，我们所看到的F_2估计算法本质上是一个线性草图算法。考虑矩阵M，其中k= $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$行，每个条目都在 $\{-1, 1\}$之间。草图简单地是 $\mathbf{z} = M\mathbf{x}$。—算法对草图进行后处理以输出其估计值。

请注意，由于草图是线性的，因此$\mathbf{x}$是否为负并不重要。实际上，从分析中也很容易看出这一点。特别地，这意味着我们可以估计 $||f_\sigma - f_{\sigma'}||_2$，其中 f_σ和 $f_{\sigma'}$分别是 σ和 σ'的频率向量。同样，如果$\mathbf{x}$，$\mathbf{x}'$是表示时间序列的两个n维信号，则它们之间的 ℓ_2范数可以通过对信号进行一次遍历来估计，即使信号是通过一系列更新给出的，这些更新甚至可以交错（当然，我们需要知道更新来自哪个信号的身份）。

3 约翰逊-林登斯特劳斯引理和维度缩减 in ℓ_2

AMS线性草图 for F_2estimation 看起来像是魔法。理解这一点的一种方法是通过
著名的约翰逊-林登斯特劳斯引理给出的对 ℓ_2空间的维度缩减
它有许多应用。JL引理可以如下表述。

定理1 (JL引理) *Let Let* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ 是 $\mathbb{R}^d$中的任意$_n points/v\mathbf{e}ctors$。对于任意 $\epsilon \in (0, 1/2)$, 存在线性映射 $f : \mathbb{R}^d \to \mathbb{R}^k$ 其中 $k \le 8\ln n/\epsilon^2$ ，使得对于所有 $1 \le i < j \le n$,

$$(1-\epsilon)||v_i - v_j||_2 \le ||f(v_i) - f(v_j)||_2 \le ||v_i - v_j||_2.$$

此外 f可以在随机多项式时间内获得。

JL引理的含义是，任意 n个点在 d维欧几里德空间中可以投影到 $O(\ln n/\epsilon^2)$维，同时保持它们之间的欧几里德距离。
证明JL引理的简单随机算法如下。令 M为一个$k \times d$矩阵，其中每个元素 Mi$_j$ 独立地从标准 $\mathcal{N}(0,1)$正态分布中选择。那么映射 f定义为 $f(\mathbf{v}) =$
$\frac{1}{\sqrt{k}}M\mathbf{v}$。现在我们简要介绍一下为什么这个算法有效。

引理2 令 $Z_1, Z_2, \ldots, Z_k$为独立的 $\mathcal{N}(0,1)$随机变量，令 $Y = \sum_{i=1}^{k} Z_i^2$。存在一个常数 c使得，

$$\Pr[(1-\epsilon)^2 k \le Y \le (1+\epsilon)^2 k] \le 2e^{c\epsilon^2 k}.$$

换句话说，k个标准正态变量的平方和集中在其均值周围。 事实上， Y的分布有一个名字，参数为k的 χ^2分布。 我们不会证明前面的引理。 可以在各个地方找到通过标准Chernoff类型论证的证明。

假设引理成立，我们可以证明以下内容。

引理3 设 $\epsilon in(0,1/2)$， $\mathbf{v}$是 $\mathbb{R}^d$中的单位向量，则 $(1-\epsilon)\leq ||M\mathbf{v}||_2 \leq (1+\epsilon)||$，概率至少为 $(1-2e^{c\epsilon^2 k})$。

证明：首先我们观察到一个关于正态分布的众所周知的事实。 设 X和 Y是独立的 $\mathcal{N}(0,1)$ 随机变量。 那么 $aX+bY$ 是 $\mathcal{N}(0, \sqrt{a^2+b^2})$ 随机变量。

设 $\mathbf{u}=\sqrt{k}M\mathbf{v}$。 注意 $\mathbf{u}$ 是一个随机向量。 注意 $u_i=\sum_{j=1}^{n} v_j\sqrt{k}M_{ij}$。 由于每个 $\sqrt{k}M_{ij}$是 $\mathcal{N}(0,1)$ 随机变量且所有元素是独立的，我们有 $u_i \simeq \mathcal{N}(0,1)$ 因为 u_i的方差是 $\sum_j v_i^2=1$ (注意 $\mathbf{v}$ 是一个单位向量)。因此 $u_1, u_2, \ldots, u_k$ 是独立的 $\mathcal{N}(0,1)$ 随机变量。 因此 $||u||^2{}_2 = \sum$ 应用引理2，我

$$\Pr[(1-\epsilon)^2 k \leq ||u||_2^2 \leq (1+\epsilon)^2 k] \geq 1-2e^{c\epsilon^2 k}.$$

□

单位向量对证明很方便，但通过缩放可以得到以下简单的推论。

推论4 设 $\epsilon in(0,1/2)$且 $\mathbf{v}$是 $\mathbb{R}^d$中的任意向量，则 $(1-\epsilon)||\mathbf{v}||_2 \leq ||M\mathbf{v}||_2 \leq (1+\epsilon)||||\mathbf{v}||_2$ 至少以概率 $(1-2e^{c\epsilon^2 k})$成立。

现在，JL引理通过并集界限很容易得到。 设 $k=c'\ln n/\epsilon^2$其中 c'基于 c选择。 考虑任意一对 $\mathbf{v}_i, \mathbf{v}_j$。

$$\Pr[(1-\epsilon)||\mathbf{v}_i-\mathbf{v}_j||_2 \leq ||M(\mathbf{v}_i-\mathbf{v}_j||_2 \leq (1+\epsilon)||\mathbf{v}_i-\mathbf{v}_j||_2] \geq (1-2e^{c\epsilon^2 k}) \geq 1-2e^{c\epsilon^2 \cdot c'\ln n/\epsilon^2} \geq 1-\frac{2}{n^{cc'}}.$$

如果 $cc' \geq 3$，则 $\mathbf{v}_i$和 $\mathbf{v}_j$之间距离的保持在相对 ϵ-近似的概率至少为$1-1/n^3$。 由于只有 $n(n-1)/2$对距离，通过并集边界，所有这些距离都被保持在这个误差容限内的概率至少为$(1-1/n)$。

参考文献：请参阅阿米特・查克拉巴蒂的讲义第6章。 算法界已经做了很多工作，以加快评估降维的速度。

一个有趣的结果是由Achlioptas [1]得出的，他表明我们选择的矩阵 M的条目可以实际上选择自 $\mathcal{N}(0,1)$中的 $\{-1,0,1\}$；离散的条目创建了一个更稀疏的矩阵，而且结果矩阵乘法更容易计算。

参考文献

[1] Dimitris Achlioptas. 数据库友好的随机投影：使用二进制硬币的Johnson-Lindenstrauss。计算机与系统科学杂志, 66(4):671–687, 2003.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. 近似频率矩阵的空间复杂度。计算机与系统科学杂志, 58(1):137–147, 1999. 初步版本在A*CM* ST*OC*1996年的会议中。

CS 598CSC: 大数据算法	讲座日期：2014年9月9日
讲师：Chandra Chekuri	记录员：Chandra Chekuri

1 用于 F_p估计的草图，其中 $0< p \leq 2$

我们已经看到了一个线性草图估计 F_2estimation，它使用 $O(\log n)$ 空间。 Indyk [1]得到了一个技术上复杂而有趣的草图，用于 $F_k$estimation，其中 $0< p \leq 2$（注意 p可以是一个实数），它使用 polylog(n) 空间。 由于细节相当复杂，我们只会给出高层次的方法，并将读者引用到论文和相关笔记中获取更多细节。 请注意，对于$p > 2$，所需空间有一个 $\Omega(n^{1-2/p})$ 的下界。

为了描述 $0< p \leq 2$ 的草图，我们将重新审视使用正态分布属性的 F_2estimate的 JL 引理方法。

F_2-估计:

让 $Y_1, Y_2, \ldots, Y_n$从 $\mathcal{N}(0,1)$ 分布中独立采样
$z \leftarrow 0$
当 (流不为空) 执行
  (i_j, Δ_j) 是当前标记
  $z \leftarrow z + \Delta_j \cdot Y_{i_j}$
结束
输出 z^2

让 $Z = \sum_{i \in [n]} x_i Y_i$是表示流结束时 z值的随机变量。 变量 Z是独立正态变量的和，根据正态分布的性质 $Z \sim \sqrt{\sum}$正态分布因此被称为$_2$稳定。 更一般地，如果以下属性成立，则分布 $\mathcal{D}$被称为 p-稳定：让 $Z_1, Z^2, \ldots, Z_n$是独立随机变量，按照 $\mathcal{D}$分布。那么 $\sum_i x_i Z_i$具有相同的分布 as $\|x\|_p Z$ where $Z \sim \mathcal{D}$. 请注意，a p-稳定分布将在0周围对称。

众所周知，对于所有的 $p \in (0,2]$，存在 p-稳定分布，但对于任何$p > 2$，不存在。一般情况下，p-稳定分布没有解析公式，除了一些特殊情况。我们已经看到，标准正态分布是2-稳定的。1-稳定分布是柯西分布，它是两个独立的标准正态随机变量的比率的分布。柯西分布的密度函数已知为 $\frac{1}{\pi(1+x^2)}$; 请注意，柯西分布没有有限的均值或方差。 我们用 $\mathcal{D}$ p 表示$_p$-稳定分布。

尽管一般的 p-稳定分布没有解析公式，但已知可以从 $\mathcal{D}$中进行采样。 Chambers-Mallows-Stuck方法如下：

- 从 $[-\pi/2, \pi/2]$ 均匀采样 θ。
- 从 $[0, 1]$ 均匀采样 r。
- 输出

$$\frac{\sin(p\theta)}{(\cos\theta)^{1/p}} \left(\frac{\cos((1-p)\theta)}{\ln(1/r)} \right)^{(1-p)/p}.$$

我们还需要一个定义。

定义**1**：分布 $\mathcal{D}$的中位数是 θ，如果对于 $Y\sim\mathcal{D}$， $\Pr[\ Y\ \leq\mu]=1/2$。 如果 $\phi(x)$是 $\mathcal{D}$的概率密度函数，则有 $\int_{-\infty}^{\mu}\phi(x)dx=1/2$。

请注意，对于一个分布来说，中位数可能没有唯一定义。 分布 $\mathcal{D}_p$有一个唯一的中位数，因此我们将使用术语中位数$(\mathcal{D}_p)$来表示这个数量。 对于一个分布 $\mathcal{D}$，我们将称之为 $|\mathcal{D}|$从 $\mathcal{D}$中随机抽取的随机变量的绝对值的分布。 如果 $\phi(x)$ 是 $\mathcal{D}$的密度函数，那么 $|\mathcal{D}|$的密度函数由 ψ给出，其中 $\psi(x)=2\phi(x)$ 如果 $x\geq0$，且 $\psi(x)=0$ 如果$x<0$。

F_p估计：

$k\leftarrow\Theta(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$
设 M是一个 $k\times n$矩阵，其中每个 $M_{ij}\sim\mathcal{D}_p$
$\mathbf{y}\leftarrow M\mathbf{x}$
输出 $Y\ \leftarrow$ 中位数$\frac{|y_1|,|y_2|,\ldots,|y_k|)}{\text{中位数}(|\mathcal{D}_p|)}$

根据 p稳定性属性，我们可以看到每个 $y_i\sim\|x\|_pY$其中$Y\sim\mathcal{D}_p$。 首先，考虑 $k=1$的情况。 然后，输出 $|y_1|/$中位数$(|\mathcal{D}_p|)$的分布符合 $c|\mathcal{D}_p|$其中$c=\|x\|_p/$中位数$(|\mathcal{D}_p|)$。 很容易验证该分布的中位数为 $\|x\|_p$。 因此，该算法从该分布中取 k个样本，并将其作为估计器的样本中位数输出。

下面的引理表明样本中位数具有良好的集中性质。

引理**1** 设$\epsilon>0$， $\mathcal{D}$是具有密度函数 ϕ和唯一中位数 $\mu>0$的分布。 假设 ϕ在 $[(1-\epsilon)\mu,(1+\epsilon)\mu]$上绝对连续，并且 $\alpha=\min\{\phi(x)\mid x\in[(1-\epsilon)\mu,(1+\epsilon)\mu]$。令 Y =中位数$(\ Y_1,Y_2,\ldots\ldots\ldots\ldots\ldots\ldots,Y_k)$其中$Y_1,\ldots,Y_k$是从分布 $\mathcal{D}$中独立取样的。那么

$$\Pr[|Y-\mu|\geq\epsilon\mu]\leq2e^{-\frac{2}{3}\epsilon^2\mu^2\alpha^2k}.$$

我们概述证明上界 $\Pr[Y\leq(1-\epsilon)\mu]$. 另一个方向类似. 注意 根据中位数的定义, $\Pr[Y_j\leq\mu]=1/2$. 因此

$$\Pr[Y_j\leq(1-\epsilon)\mu]=1/2-\int_{(1-\epsilon)\mu}^{\mu}\phi(x)dx.$$

令 $\gamma=\int$很容易看出 $\gamma\geq\alpha\epsilon\mu$.

令 I_j为指示事件 $Y_i\leq(1-\epsilon)\mu$; 我们有 $\mathbf{E}[I_j]=\Pr[Y_i\leq(1-\epsilon)\mu]\leq1/2-\gamma$.
让我 $=\sum$由于Y是$Y_1,Y2,\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$.的中位数$Y\,k,\ Y\ \leq(1-\epsilon)\mu$ 只有当 I_j中为真的个数超过 $k/2$ 时，才成立，这与 $\Pr[I>(1+\delta)\mathbf{E}[I]]$ 相同，其中 $1+\delta=\frac{1}{1-2\gamma}$.
现在，通过切诺夫界限，这个概率最多为 $e^{-\gamma^2k/3}$，对于足够小的 γ。

我们现在可以将引理应用于算法输出的估计器。 我们让 ϕ成为 $c|\mathcal{D}_p|$的分布。 回想一下，这个分布的中位数是 $\|x\|_p$，而算法的输出是从这个分布中取 k个独立样本的中位数。因此，根据引理 ，

$$\Pr[|Y-\|x\|_p|\geq\epsilon\|x\|_p]\leq2e^{-\epsilon^2k\mu^2\alpha^2/3}.$$

令 ϕ'为 $|\mathcal{D}_{\sqrt{}}|$的分布， μ'为 ϕ'的中位数。 则可以看出 $\mu\alpha = \mu'\alpha'$

其中 $\alpha' = \min\{\phi'(x) \mid (1-\epsilon)\mu' \leq (1+\epsilon)\mu'\}$. 因此 $\mu'\alpha'$仅取决于 $\mathcal{D}_p$和 ϵ. 将此表示为 $c_{p,\epsilon}$，我们有，

$$\Pr[|Y - \|x\|_p| \geq \epsilon\|x\|_p] \leq 2e^{-\epsilon^2 k c_{p,\epsilon}^2/3} \leq (1-\delta),$$

给定 $k = \Omega(c_{p,\epsilon} \cdot \frac{1}{\epsilon^2}\log\frac{1}{\delta})$.

技术问题：在前面的描述中，有几个技术问题需要解决才能得到一个合适的算法。 首先，按照描述的算法需要存储整个矩阵 M，这对于流式应用来说太大了。 其次，常数 k取决于 $c_{p,\epsilon}$，由于一般情况下 $\mathcal{D}_p$对于一般的 p不是很清楚，所以$\mathbf{c}_{\mathrm{p},\square}$也不是明确的。 为了得到一个流式算法，非常高层次的想法是通过使用N**isan**的小空间伪随机生成器来去随机化算法。更多细节请参见[1]。

2 统计频繁项

我们已经看到了各种算法来估计不同的 $F_p$norms，其中 $p \geq 0$。 注意， F_0对应于不同元素的数量。 在极限情况下，当 $p \to \infty$时，向量 $\mathbf{x}$的 $\ell_p$norm是 $\mathbf{x}$的绝对值的最大值。 因此，我们可以将 F_∞norm定义为找到 $\mathbf{x}$中的最大频率。 更一般地，我们希望在流中找到频繁项，也称为“重要项”。 一般来说，如果相对于 m来说太小，那么用有限的空间估计最重的频率是不可行的。

2.1 Misra-Greis算法用于频繁项

假设我们有一个流 $\sigma = a_1, a_2, \ldots, a_m$其中 $a_j \in [n]$，简单的设置，我们想要找到所有满足 $f_i > m/k$的元素。 请注意，最多可能有 k个这样的元素。 最简单的情况是当 $k = 2$时，我们想知道是否存在一个“多数”元素。 对于 $k = 2$，有一个简单的确定性算法，也许你们在算法课上都见过。

该算法使用大小为 k的关联数组数据结构。

MisraGreis(k):

D是一个空的关联数组
当 (流不为空) 时
  a_j 是当前项
  如果 (a_j在D的键中)
    $D[a_j] \leftarrow D[a_j] + 1$
  否则如果 (|D的键|$< k-1$) 则
    $D[a_j] \leftarrow 1$
  否则
    对于每个 $\ell \in$键(D) do
      $D[\ell] \leftarrow D[\ell] - 1$
    从 D中删除计数值为0的元素
  endWhile
  对于每个 $i \in$键(D) 设置$\hat{f}_i = D[i]$
  对于每个 $i \in$键(D) 设置$\hat{f}_i = 0$

我们将以下内容留给读者作为练习。

引理**2** 对于每个 $i \in [n]$ *:*

$$f_i - \frac{m}{k} \leq \hat{f}_i \leq f_i.$$

该引理意味着如果 $f_i > m/k$那么 $i \in$键(D) 在算法结束时。 因此，可以使用第二次遍历数据来计算 $keys(D)$ 中 k个元素的确切 f_i值。 这为找到所有频率至少为 m/k的项提供了一个 $O(kn)$ 时间的两次遍历算法。

参考文献：有关当 $0 < p \leq 2$ 时的估计更多细节，请参阅 Indyk [1] 的原始论文，Am**it** C**ha**k**rabarti** 的笔记（第7章）和 **Jela**n**i Nelso**n 的课程第4讲。

参考文献

[1] Piotr Indyk. 稳定分布，伪随机生成器，嵌入和数据流计算。*ACM*期刊（*JACM*），53（3）：307-323，2006年。

CS 598CSC: 大数据算法	讲座日期：2014年9月11日
讲师：Chandra Chekuri	记录员：Chandra Chekuri

Misra-Greis确定性计数保证可以使用 $O(k)$ 计数器和更新时间 $O(\log k)$ 找到所有频率$> F_1/k$的项。 设置 $k= 1/\epsilon$可以将算法视为为每个 f_i提供一个加法 ϵF_1近似。 然而，该算法不提供草图。 线性草图算法的一个优点是能够处理删除操作。 我们现在讨论了两个具有多个应用的草图算法。 这些草图可以用于估计点查询：在看到一个流 σover $[n]$ 中的项后，我们希望估计$_i \in [n]$ 的频率 $f\ i$。 更一般地，在转门模型中，我们希望估计给定$_i \in [n]$ 的 $x\ i$。 我们只能保证估计值具有加法误差。

1 CountMin Sketch

我们首先描述更简单的CountMin草图。 草图维护了几个计数器。 计数器最好被视为一个宽度为 w 和深度为 d的矩形数组。 对于每一行 i，我们有一个哈希函数 h_i: $[n] \to [w]$，将元素映射到 w个桶之一。

CountMin-Sketch(w, d):

$h_1, h_2, \ldots, h_d$是从$[n] \to [w]$的两两独立的哈希函数。
当（流不为空）时，执行
  $a_t=$ 当前项（i_t，Δ_t）
  对于 ℓ从1到 d的每个值
    $C[\ell, h_\ell(i_j)] \leftarrow C[\ell, h_\ell(i_j)] + \Delta_t$
endWhile
对于 $i \in [n]$，设置 $\tilde{x}_i= \min^d_{\ell=1} C[\ell, h_\ell(i)]$.

计数器 $C[\ell, j]$ 简单地计算所有 x_i的总和，使得 $h_\ell(i) = j$。也就是说，

$$C[\ell, j] = \sum_{i:h_\ell(i)=j} x_{i\circ}$$

练习：CountMin是一个线性草图。 投影矩阵的条目是什么？

我们将在严格的转门模型中分析草图，其中对于所有$_i \in [n]$， $x\ i \geq 0$；请注意 Δ_t 我们将是负数。

引理**1**令 $d= \Omega(\log \frac{1}{\delta})$和$w > \frac{2}{\epsilon}$。 那么对于任意固定的 $i \in [n]$， $x_i \leq \tilde{x}_i$和

$$\Pr[\tilde{x}_i \geq x_i + \epsilon\|\mathbf{x}\|_1] \leq \delta.$$

证明:固定 $i \in [n]$. 令 $Z_\ell = C[\ell, h_\ell(i)]$ 是行 ℓ中计数器的值，其中 i被哈希到。我们有

$$\mathbf{E}[Z_\ell] = x_i + \sum_{i'=i} \Pr[h_\ell(i') = h_\ell(i)]x_{i'} = x_i + \sum_{i'=i} \frac{1}{w} x_{i'} \leq x_i + \frac{\epsilon}{2}\|\mathbf{x}\|_1.$$

注意我们使用了 h_ℓ的成对独立性来得出结论，即$\Pr[h_\ell(i') = h_\ell(i)] = 1/w$.

根据马尔可夫不等式（这里我们使用非负性质 $\mathbf{x}$），

$$\Pr[Z_\ell > x_i + \epsilon\|\mathbf{x}\|_1] \leq 1/2.$$

因此

$$\Pr[\min_\ell Z_\ell > x_i + \epsilon\|\mathbf{x}\|_1] \leq 1/2^d \leq \delta.$$

□

备注：通过选择 $\delta = \Omega(\log n)$，我们可以确保至少以概率 $(1 - 1/\text{poly}(n))$，对于所有i∈[n]，有 $\tilde{x}_i - x_i \leq \epsilon\|\mathbf{x}\|_1$。

练习：对于一般的转门流，其中 $\mathbf{x}$可以有负数条目，我们可以取计数器的中位数。 对于这个估计，你应该能够证明以下内容。

$$\Pr[|\tilde{x}_i - x_i| \geq 3\epsilon\|\mathbf{x}\|_1] \leq \delta^{1/4}.$$

2 计数草图

现在我们讨论与计数草图密切相关的计数草图，它也通过宽度 w和深度 d参数化的计数器数组来维护。

计数草图 (w, d):

$h_1, h_2, \ldots, h_d$是从$[n] \to [w]$的两两独立的哈希函数。
$g_1, g_2, \ldots, g_d$是从 $[n] \to \{-1, 1\}$的两两独立的哈希函数。
　当（流不为空）时，执行
　a_t= 当前项（i_t， Δ_t）
　对于 ℓ从1到 d的每个值
　　$C[\ell, h_\ell(i_j)] \leftarrow C[\ell, h_\ell(i_j)] + g(i_t)\Delta_t$
endWhile
对于 $i \in [n]$，设置 $\tilde{x}_i$= 中位数$\{g_1(i)C[1, h_1(i)], g_2(i)C[2, h_2(i), \ldots, g_d(i)C[d, h_d(i)]\}$.

练习：CountMin是一个线性草图。 投影矩阵的条目是什么?

引理2 设 $d \geq \log \frac{1}{\delta}$ 且w-> $\frac{3}{\epsilon^2}$. 那么对于任意固定的 $i \in [n]$， $\mathbf{E}[\tilde{x}_i] = x_i$且

$$\Pr[|\tilde{x}_i - x_i| \geq \epsilon\|\mathbf{x}\|_2] \leq \delta.$$

证明：固定一个 $i \in [n]$. 对于$i' \in [n]$，令 $Yi\prime$为指示随机变量如果 $h_\ell(\ i) = h_\ell(_{i'})$ 则$Y\tilde{i}'$为1；即i和 $i\prime$在 h_ℓ中碰撞。 注意 $\mathbf{E}\ [Y\, i] = \mathbf{E}[\mathsf{Y}_i{}^{2\prime}] = 1/w$根据 h_ℓ的两两独立性。我们有

$$Z_\ell = g_\ell(i)C[\ell, h_\ell(i)] = g_\ell(i)\sum_{i'} g_\ell(i')x_{i'}Y_{i'}$$

因此,

$$\mathbf{E}[Z_\ell] = x_i + \sum_{i'=i} \mathbf{E}[g_\ell(i)g_\ell(i')Y_{i'}]x_{i'} = x_i,$$

因为 $\mathbf{E}[g_\ell(i)g_\ell(i')] = 0$ 对于 $i = i'$由于 g_ℓ和 $Y_{i'}$的两两独立性，Yi'独立于 g_ℓ(i)和 $g_\ell(i')$. 现在我们上界估计 Z_ℓ的方差.

$$\begin{aligned}
\mathbf{Var}[Z_\ell] &= \mathbf{E}\left[(\sum_{i'=i} g_\ell(i)g_\ell(i')Y_{i'}x_{i'})^2\right] \\
&= \mathbf{E}\left[\sum_{i'=i} x_{i'}^2 Y_{i'}^2 + \sum_{i'=i''} x_{i'}x_{i''}g_\ell(i')g_\ell(i'')Y_{i'}Y_{i''}\right] \\
&= \sum_{i'=i} x_{i'}^2 \mathbf{E}[Y_{i'}^2] \\
&\leq \|\mathbf{x}\|_2^2/w.
\end{aligned}$$

使用切比雪夫,

$$\Pr[|Z_\ell - x_i| \geq \epsilon\|\mathbf{x}\|_2] \leq \frac{\mathbf{Var}[Z_\ell]}{\epsilon^2\|\mathbf{x}\|_2^2} \leq \frac{1}{\epsilon^2 w} \leq 1/3.$$

现在，通过切尔诺夫界限,

$$\Pr\left[|\text{median}\{Z_1, \ldots, z_d\} - x_i| \geq \epsilon\|\mathbf{x}\|_2\right] \leq e^{-cd} \leq \delta.$$

因此选择 $d = O(\log n)$ 并且取中位数可以保证所需的边界，概率很高。

□

备注：通过选择 $\delta = \Omega(\log n)$，我们可以确保至少有$(1 - 1/\text{poly}(n))$的概率满足 $|\tilde{x}_i - x_i| \leq \epsilon\|\mathbf{x}\|_2$对于所有 $i \in [n]$。

3个应用

Count和CountMin草图已经找到了许多应用。 请注意，它们具有类似的结构，尽管保证不同。 考虑估计频率矩的问题。 Count草图输出一个估计值$\tilde{f}_i$ for f_i，其附加误差为 $\epsilon\|\mathbf{f}\|_2$，而CountMin保证了一个附加误差为 $\epsilon\|\mathbf{f}\|_1$，这总是更大的。 当 $\mathbf{x} \geq 0$时，CountMin提供了单边误差，这有一些好处。 CountMin使用 $O(\frac{1}{\epsilon}\log\frac{1}{\delta})$个计数器，而Count草图使用$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$个计数器。 请注意，Misra-Greis算法使用 $O(1/\epsilon)$-计数器。

3.1 重要元素

如果对于所有的索引 i，满足 $x_i \geq \alpha\|x\|_1$，我们将称其为 α-HH（重要元素）。我们希望找到所有 α重要元素的集合 S_α。我们将放宽这个假设，输出一个集合 S，使得

$$S_\alpha \subseteq S \subseteq S_{\alpha-\epsilon}。$$

在这里，我们假设$\alpha < \alpha$，否则近似结果没有意义。

假设我们使用了CountMin草图，其中 $w = 2/\epsilon$， $\delta = c/n$，对于足够大的 c。 然后，正如我们所见，至少有$(1 - 1/\text{poly}(n))$的概率，对于所有的 $i \in [n]$，

$$x_i \leq \tilde{x}_i \leq x_i + \epsilon\|\mathbf{x}\|_1。$$

一旦计算出草图，我们只需遍历所有 i并将 i添加到 S中，如果 $\tilde{x}_i \geq \alpha\|\mathbf{x}\|_1$。 很容易看出 S是所需的集合。

不幸的是，计算 S是昂贵的。 草图具有 $O(\frac{1}{\epsilon}\log n)$ 计数器，并且处理每个 i的时间与计数器的数量成比例，因此总时间为$O(\frac{1}{\epsilon}n\log n)$ 以输出大小为 $O(\frac{1}{\alpha})$ 的集合 S。 事实证明，通过以分层方式在草图中保留附加信息，可以将时间削减为与 $O(\frac{1}{\alpha}\text{polylog}(n)))$ 成比例。

3.2 范围查询

在几个应用中，范围 $[n]$ 对应于项目的实际总排序。 例如，$[n]$ 可以表示时间的离散化， $\mathbf{x}$对应于信号。在数据库中，$[n]$ 可以表示有序的数值属性，例如人的年龄、身高或工资。 在这种情况下，范围查询非常有用。 范围查询是形式为 $[i,j]$ 的区间，其中$i,j \in [n]$，且$i \leq j$。目标是输出 $\sum$

请注意，有 $O(n^2)$个潜在的

有一个简单的技巧可以使用我们所见过的草图来解决这个问题。 如果 $j-i+1$是2^k并且2^k可以整除 $i-1$，则区间$[i,j]$是一个二进制区间/范围。 假设 n是2的幂。 那么长度为1的二进制区间是$[1,1],[2,2],\ldots,[n,n]$。 长度为2的二进制区间是$[1,2],[3,4],\ldots$长度为4的二进制区间是$[1,4],[5,8],\ldots$。

主张**3** 每个区间 $[i,j]$可以表示为至多$2\log n$个二进制区间的不相交并集。

因此，维护二进制范围的准确点查询就足够了。 注意，最多有2个二进制范围。 它们根据长度分为 $O(\log n)$组；给定长度的范围将整个区间划分为若干个二进制间隔。我们可以为长度为i的$n/2^i$二进制间隔保留一个单独的CountMin草图（其中$i=0$对应于点查询的草图）。 使用这些 $O(\log n)$个CountMin草图，我们可以回答任何范围查询，并具有添加误差 $\epsilon\|\mathbf{x}\|_1$。 注意，一个范围 $[i,j]$可以表示为2个$\log n$点查询的总和，每个查询都具有添加误差。 因此，为了确保范围查询的添加误差为$\epsilon\|\mathbf{x}\|_1$，必须选择草图的 ϵ'为 $\epsilon/(2\log n)$。

通过选择 $d = O(\log n)$，所有点查询在所有草图中的错误概率将不超过$1/\text{poly}(n)$。 这将确保所有范围查询的答案在添加$\epsilon\|\mathbf{x}\|_1$内。 总空间将为 $O(\frac{1}{\epsilon}\log^3 n)$

3.3 稀疏恢复

设 $\mathbf{x} \in \mathbb{R}^n$为一个向量。 我们能否通过一个稀疏向量 $\mathbf{z}$来近似 $\mathbf{x}$？ 稀疏意味着 $\mathbf{z}$最多有 k个非零条目，其中 k是给定的某个值（这与 $\|\mathbf{z}\|_0 \leq k$相同）。 一个合理的模型是要求计算误差

$$\text{err}_p^k(\mathbf{x}) = \min_{\mathbf{z}:\|\mathbf{z}\|_0 \leq k} \|\mathbf{x}-\mathbf{z}\|_p$$

对于一些 p. 一个典型的选择是 $p=2$. 很容易看出，通过限制$\mathbf{x}$到它的 k个最大坐标（绝对值）可以获得最优 $\mathbf{z}$. 我们在这里提出的问题是我们是否可以以流式方式高效地估计 $\text{err}_2^k(\mathbf{x})$. 为此，我们使用Count sketch. 回想一下，通过选择 $w=3/\epsilon^2$和 $d=\Theta(\log n)$，该草图可以高概率地保证

$$\forall i \in [n], \quad |\tilde{x}_i - x_i| \leq \epsilon\|\mathbf{x}\|_2.$$

事实上，我们可以展示一个推广.

引理**4** *Count-Sketch with* $w = 3k/\epsilon$ *and* $d = O(\log n)$ 确保

$$\forall i \in [n], \quad |\tilde{x}_i - x_i| \leq \frac{\epsilon}{\sqrt{k}} err_2^k(\mathbf{x}).$$

证明：设 $S = \{i_1, i_2, \ldots, i_k\}$为 $\mathbf{x}$中最大坐标的索引集合，设 $\mathbf{x}'$为通过将 $\mathbf{x}$中索引集合 S中的元素置零得到的新向量。 注意，$\mathbf{err}_2^k(\mathbf{x}) = \|\mathbf{x}'\|_2$。 固定一个坐标$i$。 考虑第 ℓ行，设 $Z_\ell = g_\ell(i)C[\ell, h_\ell(i)]$，与之前一样。 设 A_ℓ为存在一个索引 $t \in S$使得 $h_\ell(i) = h_\ell(t)$的事件，即任何"大"坐标在 h_ℓ下与 i发生碰撞。注意，$\mathbf{Pr}[A_\ell] \leq \sum_{t \in S} \Pr[h_\ell(i) = h_\ell(t)] \leq |S|/w \leq \epsilon/3$ 通过成对独立性估计

$$\begin{aligned}
\Pr[|Z_\ell - x_i| \geq \frac{\epsilon}{\sqrt{k}} \mathrm{err}_2^k(\mathbf{x})] &= \Pr[|Z_\ell - x_i| \geq \frac{\epsilon}{\sqrt{k}} \|\mathbf{x}'\|_2] \\
&= \Pr[A_\ell] \cdot \Pr[|Z_\ell - x_i| \geq \frac{\epsilon}{\sqrt{k}} \|\mathbf{x}'\|_2] + \Pr[|Z_\ell - x_i| \geq \frac{\epsilon}{\sqrt{k}} \|\mathbf{x}'\|_2 \mid \neg A_\ell] \\
&\leq \Pr[A_\ell] + 1/3 < 1/2.
\end{aligned}$$

□

现在让$\tilde{\mathbf{x}}$成为从草图中获得的对 $\mathbf{x}$的近似。 我们可以取$\tilde{\mathbf{x}}$的 k个最大坐标来形成向量 $\mathbf{z}$并输出 $\mathbf{z}$。 我们声称这给出了对误差$\mathrm{err}_2^k(\mathbf{x})$的良好近似。 为了证明这一点，我们证明以下引理。

引理**5** 设 $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ 满足

$$\|\mathbf{x} - \mathbf{y}\|_\infty \leq \frac{\epsilon}{\sqrt{k}} err_2^k(\mathbf{x}).$$

那么，

$$\|\mathbf{x} - \mathbf{z}\|_2 \leq (1 + 5\epsilon) err_2^k(\mathbf{x}),$$

其中 $\mathbf{z}$是按照以下方式获得的向量： $\mathbf{z}_i = \mathbf{y}_i$对于 $i \in T$其中 T是 $\mathbf{y}$的 k个最大（绝对值）索引的集合， $\mathbf{z}_i = 0$对于 $i \in T$。证明：设 $t = \frac{1}{\sqrt{k}} \mathrm{err}_2^k(\mathbf{x})$ 以帮助简化符号。 令 S为坐标最大的索引集 of $\mathbf{x}$. 我们有,

$$(\mathrm{err}_2^k(\mathbf{x}))^2 = kt^2 = \sum_{i \in [n] \setminus S} x_i^2 = \sum_{i \in T \setminus S} x_i^2 + \sum_{i \in [n] \setminus (S \cup T)} x_i^2.$$

我们写成:

$$\begin{aligned}
\|\mathbf{x} - \mathbf{z}\|_2^2 &= \sum_{i \in T} |x_i - z_i|^2 + \sum_{i \in S \setminus T} |x_i - z_i|^2 + \sum_{i \in [n] \setminus (S \cup T)} x_i^2 \\
&= \sum_{i \in T} |x_i - y_i|^2 + \sum_{i \in S \setminus T} x_i^2 + \sum_{i \in [n] \setminus (S \cup T)} x_i^2.
\end{aligned}$$

我们分别处理每个项。 第一个项很容易界定。

$$\sum_{i \in T} |x_i - y_i|^2 \leq \sum_{i \in T} \epsilon^2 t^2 \leq \epsilon^2 k t^2.$$

第三项是 $\|\mathbf{x}-\mathbf{z}\|_2$和 $\mathrm{err}_2^k(\mathbf{x})$ 的共同项。 第二项是需要关注的项。

注意 S是 $\mathbf{x}$中最大的 k个坐标的集合， T是 $\mathbf{y}$中最大的 k个坐标的集合。 因此$|S \setminus T| = |T \setminus S|$ ，称它们的基数为 $\ell \geq 1$。 由于 $\mathbf{x}$和 $\mathbf{y}$在 ℓ_∞范数中接近（即它们在每个坐标上接近），这意味着 $S \setminus T$和 $T \setminus S$中的坐标在 $\mathbf{x}$中大致具有相同的值。 更准确地说，令 $a = \max_{i \in S \setminus T} |x_i|$和 $b = \min_{i \in T \setminus S} |x_i|$。 我们将其留给读者作为一个练习，来证明 $a \leq b + 2\epsilon t$，因为 $\|\mathbf{x}-\mathbf{y}\|_\infty \leq \epsilon t$。

因此,

$$\sum_{i \in S \setminus T} x_i^2 \leq \ell a^2 \leq \ell(b + 2\epsilon t)^2 \leq \ell b^2 + 4\epsilon k t b + 4k\epsilon^2 t^2.$$

但是我们有

$$\sum_{i \in T \setminus S} x_i^2 \geq \ell b^2.$$

将事物放在一起,

$$\begin{aligned} \|\mathbf{x}-\mathbf{z}\|_2^2 &\leq \ell b^2 + 4\epsilon k t b + \sum_{i \in [n] \setminus (S \cup T)} x_i^2 + 5k\epsilon^2 t^2 \\ &\leq \sum_{i \in T \setminus S} x_i^2 + \sum_{i \in [n] \setminus (S \cup T)} x_i^2 + 4\epsilon(\mathrm{err}_2^k(\mathbf{x}))^2 + 5\epsilon^2(\mathrm{err}_2^k(\mathbf{x}))^2 \\ &\leq (\mathrm{err}_2^k(\mathbf{x}))^2 + 9\epsilon(\mathrm{err}_2^k(\mathbf{x}))^2. \end{aligned}$$

引理的证明是因为对于足够小的 ϵ， $\sqrt{1 + 9\epsilon} \leq 1 + 5\epsilon$。

□

参考文献注释：Count sketch 是由 Charikar、Chen 和 Farach-Colton [1] 提出的。 CountMins ketch 是由 Cormode 和 Muthukrishnan [4] 提出的；有关多个应用的论文请参阅这些论文。 Cormode 在 [2] 中对草图技术进行了很好的概述。 有关查找频繁项的算法的比较分析（理论和实验）请参阅 [3]。 一种确定性的 CountMin 变体称为 CR-Precis 很有趣；请参阅http://polylogblog.wordpress.com/2009/09/22/bite-sized-streams-cr-precis/以获取有关指针和一些评论的博文。 这些应用程序摘自 McGregor 和 Muthukrishnan 的草稿书的第一章。

参考文献

[1] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 在数据流中找到频繁项理论计算机科学, 312(1):3–15, 2004.

[2] Graham Cormode, Minos N. Garofalakis, Peter J. Haas, and Chris Jermaine. 大规模数据的概要: 样本, 直方图, 小波, 轮廓数据库的基础与趋势,4(1-3):1–294, 2012.

[3] Graham Cormode and Marios Hadjieleftheriou. 在数据流中找到频繁项的方法 *VLDB J.*, 19(1):3–20, 2010.

[4] Graham Cormode and S. Muthukrishnan. 一种改进的数据流摘要: 计数最小草图及其应用*J. Algorithms*, 55(1):58–75, 2005.

CS 598CSC: 大数据算法	讲座日期：2014年9月23日
讲师：Chandra Chekuri	记录员：Chandra Chekuri

1 优先采样和求和查询

假设我们有一个流 $a_1, a_2, \ldots, a_n$(是的，我们在这里从 m更改符号为 n表示流的长度) 的对象，每个 a_i都有一个非负权重 w_i. 我们想要存储一个代表性样本 $S \subset [n]$ 的项目，以便我们可以回答子集求和查询。 也就是说，给定一个查询 $I \subseteq [n]$，我们想要回答 $\sum$ 一种方法是如下所示。 假设我们选择 S如下：独立地对每个 $i \in [n]$ 进行采样，概率为 p_i，如果选择了 i，我们设置一个缩放权重 $\hat{w}_i = w_i / p_i$. 现在，给定一个查询 I，我们输出其权重的估计值为 $\sum_{i \in I \cap S} \hat{w}_i$. 注意估计的期望等于 $w(I)$. 这种方案的缺点主要与我们无法控制样本大小有关。 这意味着我们无法充分利用可用的内存。 与第一个相关的是，如果我们事先不知道流的长度，即使我们愿意在内存大小上灵活，我们也无法确定采样率。 Duffield、Lund和Thorup提出的一种优雅的方案，称为优先采样，克服了这些限制。 他们考虑了给定参数 k的情况下，维护一个 k-样本 S以及一些权重 $\hat{w}_i$，其中 $i \in S$，以便我们可以回答子集和查询。

他们的方案如下，描述为如果 a_1，a_2，. . . ，a_n是离线可用的。

1. 对于每个 $i \in [n]$，设置优先级 $q_i = w_i/u_i$其中 u_i是从[0,1]中随机选择的均匀分布的（与其他项目独立选择）.

2. S是具有 $k$highest优先级的项目集合.

3. τ是第$(k+1)$高的优先级. 如果 $k \geq n$，则设置 $\tau = 0$.

4. 如果 $i \in S$，则设置 $\hat{w}_i = \max\{w_i, \tau\}$，否则设置$\hat{w}_i = 0$.

我们观察到上述抽样可以通过简单地保持当前样本 S和当前阈值 τ来在流式计算环境中实现. 我们将其作为一个练习来展示当新项目到达时如何更新这些信息.

我们展示了一些优先级抽样的有趣且非显而易见的特性. 我们将假设简单起见，$1 < k < n$. 第一个是我们想要的基本特性.

引理 **1** $\mathbf{E}[\hat{w}_i] = w_i$.

证明:固定 i. 令 $A(\tau')$ 为事件，其中 k'th 最高优先级的项 $j = i$是 τ'. 注意如果 $i \in S$且 $q_i = w_i/u_i \geq \tau'$，那么 $i \in S$，则 $\hat{w}_i = \max\{w_i, \tau'\}$，否则 $\hat{w}_i = 0$. 为了评估 $\Pr[i \in S \mid A(\tau')]$，我们考虑两种情况。

情况 1: $w_i \geq \tau'$. 在这种情况下，我们有 $\Pr[i \in S \mid A(\tau')] = 1$ 且$\hat{w}_i = w_i$。

情况 2: $w_i < \tau'$. 那么 $\Pr[i \in S \mid A(\tau')] = \frac{w_i}{\tau'}$ 和 $\hat{w}_i = \tau'$.

在这两种情况下，我们可以看到 $\mathbf{E}[\hat{w}_i] = w_i$. □

前面的论述表明估计量 $\sum_{\cap S} \hat{w}_i$的期望等于 $w(I)$. 我们还可以通过阈值 τ来估计$\hat{w}_i$的方差.

引理**2** 方差 $[\hat{w}_i] = \mathbf{E}[\hat{v}_i]$ 其中 $\hat{v}_i = \begin{cases} \tau \max\{0, \tau - w_i\} & if\ i \in S \\ 0 & 如果\ i \in S \end{cases}$

证明**:**固定 i. 我们定义 $A(\tau')$ 为事件 τ'是元素 $j = i$中第k高的优先级. 证明基于展示

$$\mathbf{E}[\hat{v}_i \mid A(\tau')] = \mathbf{E}[\hat{w}_i^2 \mid A(\tau')] - w_i^2.$$

从前面引理的证明概要中，我们估计了左手边的值为

$$\begin{aligned} \mathbf{E}[\hat{v}_i \mid A(\tau')] &= \Pr[i \in S \mid A(\tau')] \times \mathbf{E}[\hat{v}_i \mid i \in S \wedge A(\tau')] \\ &= \min\{1, w_i/\tau'\} \times \tau' \max\{0, \tau' - w_i\} \\ &= \max\{0, w_i\tau' - w_i^2\}. \end{aligned}$$

现在我们分析右手边,

$$\begin{aligned} \mathbf{E}[\hat{w}_i^2 \mid A(\tau')] &= \Pr[i \in S \mid A(\tau')] \times \mathbf{E}[\hat{w}_i^2 \mid i \in S \wedge A(\tau')] \\ &= \min\{1, w_i/\tau'\} \times (\max\{w_i, \tau'\})^2 \\ &= \max\{w_i^2, w_i\tau'\}. \end{aligned}$$

□

令人惊讶的是，如果 $k \geq 2$，则对于任何$_i =_j$，$\hat{w}$ i和$\hat{w}$ j之间的协方差等于0。

引理**3** $\mathbf{E}[\,\hat{w}_i\hat{w}_j] = 0$。

事实上，前面的引理是下面更一般的引理的特例。

引理**4** $\mathbf{E}$ $[\prod_{i\in I} \hat{w}_i] = \prod_{i=1}^{k} w_i$如果 $|I| \leq k$，则为1，如果 $|I| > k$，则为0。

证明：很容易看出，如果 $|I| > k$，则乘积为0，因为至少有一个不在样本中。 现在我们假设 $|I| \leq k$ ，并通过对 $|I|$进行归纳来证明所需的结论。 事实上，我们需要一个更强的假设。 令 τ''为项 $j = i$ 中的第$(k - |I| + 1)$个最高优先级。 我们将以 τ''为条件，并证明 $\mathbf{E}[\prod_{i\in I} \hat{w}_i \mid A(\tau'')] = \prod$ 对于基本情况，我们已经看到了证明 $|_I| = 1$的证据。

情况1：存在一个i $\in I$使得 wi$>\tau''$。 那么显然i $\in S$且$\hat{w}$i $= w$i。在这种情况下，$\mathbf{E}[$

$$\prod_{i\in I} \hat{w}_i \mid A(\tau'')] = w_{\mathrm{i}} \cdot \mathbf{E}[\prod_{i\in I\setminus\{\mathrm{i}\}} \hat{w}_i \mid A(\tau'')],$$

我们对 $I\setminus\{\mathrm{i}\}$应用归纳法。 从技术上讲，术语$\mathbf{E}[\prod_{i\in I'} \hat{w}_i \mid A(\tau'')]$指的是 τ''是 $k - |I'| + 1$的最高优先级，其中 $I' = I\setminus\{\mathrm{i}\}$。

案例2：对于所有的 $h \in I$，都有 $w_h < \tau''$。 让 q是 I中项目的最小优先级。 如果$q < \tau''$那么对于某个$_j \in I$，$\hat{w}$ $j = 0$，并且整个乘积为0。 因此，在这种情况下，对期望没有贡献。 因此，我们将考虑 $q \geq \tau''$的情况。 这个事件的概率是 $\prod_{i\in I} \frac{w_i}{\tau''}$。 但在这种情况下，所有的 $i \in I$都将在 S中，并且此外，对于每个$_i, \hat{w}$ $i = \tau''$。 因此，$\prod$ 的期望值 $\prod_{i\in I} \hat{w}_i = \prod_{i\in I} w_i$ ，如所需。 □

结合引理2和3，估计量 $\sum$的方差 $_{i\in I\cap S} \hat{w}_i$是

$$\text{变量}[\sum_{i\in I\cap S} \hat{w}_i = \sum_{i\in I\cap S} \text{变量}[\,\hat{w}_i] = \sum_{i\in I\cap S} \mathbf{E}[\hat{v}_i].$$

这样做的好处是可以通过检查 τ和 $S \cap I$中元素的权重来计算估计量的方差。

2 ℓ_0 采样

我们已经在流式设置中看到了 ℓ_2 采样。 这些想法可以推广到 ℓ_p 采样到 ℓ_p 采样，其中 $p \in (0,2)$ — 参见[]。 然而， ℓ_0采样需要稍微不同的想法。
ℓ_0采样意味着我们从流中的不同元素中近似均匀地采样。
令人惊讶的是，即使在旋转门设置中，我们也可以做到这一点。

回想一下，我们看到的Count-Sketch的一个应用是 ℓ_2-稀疏恢复。 特别地，我们可以使用 $O(k \log n/\epsilon)$个字来获得对$\mathrm{err}_2^k(\mathbf{x})$的$(1+\epsilon)$-近似，且高概率成立。 假设 $\mathbf{x}$是 k-稀疏的，那么$\mathrm{err}_2^k(\mathbf{x}) = 0$ ！ 这意味着我们可以高概率地检测出 $\mathbf{x}$是否是 k-稀疏的，并且实际上可以识别出 $\mathbf{x}$的非零坐标。 事实上，我们可以证明以下更强的版本。

引理**5** 对于 $1 \leq k \leq n$和 $k' = O(k)$，存在一个从$O(k \log^n)$个随机比特生成的草图 $L : \mathbb{R}\, n \to \mathbb{R}k'$ ，以及一个恢复过程，该过程在输入 $L(\mathbf{x})$时具有以下特点：*(i)* 如果 $\mathbf{x}$是 k-稀疏的，则以概率 1输出 $\mathbf{x}^{'} = \mathbf{x}$，*(ii)* 如果 $\mathbf{x}$不是 k-稀疏的，则算法可以高概率地检测到这一点。

我们将使用上述内容进行 ℓ_0采样，如下所示。 我们首先描述一个高级算法，该算法不适用于流式计算，并且稍后会说明如何使其成为流式实现。

1. 对于 $h = 1, \ldots, \lfloor \log n \rfloor$，让 I_h成为 $[n]$ 的随机子集，其中 I_j has 基数 2_j。让 $I_0 = [n]$。
2. 让 $k = \lceil 4 \log(1/\delta) \rceil$。对于 $h = 0, \ldots, \lfloor \log n \rfloor$，对 $\mathbf{x}$的坐标进行 k-稀疏恢复，限制为 I_h的坐标。
3. 如果任何一个稀疏恢复成功，则输出第一个成功的稀疏恢复的随机坐标。
4. 如果没有稀疏恢复输出有效向量，则算法失败。

令 J为 $\mathbf{x}$的非零坐标集合。 我们现在证明，算法在概率 $(1-\delta)$ 下能够成功输出一个从 J中均匀采样的样本。假设 $|J| \leq k$。 那么当 $h = 0$ 时， $\mathbf{x}$被完全恢复，并且算法从 J中输出一个均匀采样。 假设 $|J| > k$。我们观察到 $\mathbf{E}[|I\, h \cap J|] = 2\, h|J|/n$，因此存在一个 h^*，使得 $\mathbf{E}[|I_{h^*} \cap J|] = 2\, h^*|J|/n$在 $k/3$ 和 $2k/3$ 之间。 通过切尔诺夫边界，我们可以证明在概率至少 $(1-\delta)$ 下，有$1 \leq |I^h{}_* \cap J| \leq k$。 对于这个 h^*，稀疏恢复将成功，并输出J的一个随机坐标。 具体的声明如下：

- 以至少$(1-\delta)$的概率，算法输出一个坐标 $i \in [n]$。
- 如果算法输出一个坐标 i，则它不是均匀随机样本的概率是因为稀疏恢复算法对某些 h失败了；我们可以使这个概率小于$1/n^c$，其中 c是任意常数。

因此，实际上我们得到了零误差 ℓ_0 样本。

如上所述，该算法需要对 I_h进行采样和存储，其中 $h = 0, \ldots, \lfloor \log n \rfloor$。 为了避免这个问题，我们可以使用Nisan的小空间计算伪随机生成器。 我们跳过这部分的细节；请参考[2]。 上述过程的总空间需求可以被证明。

以 $O(\log^2 n \log(1/\delta)$ 的错误概率为 $\delta + O(1/n^c)$ 进行估计。这在常数 δ下是近乎最优的，如[2]所示。

参考文献注释：优先采样的材料直接来自于[1]，该文献描述了应用、与先前采样技术的关系，并进行了实验评估。
优先采样在强意义上被证明是“最优的”；参见[3]。

我们描述的采样算法来自于Jowhari、Saglam和Tardos的论文[2]。
在McGregor-Muthu的草稿书的信号章节中可以看到一个更简单的算法。

参考文献

[1] Nick Duffield, Carsten Lund, and Mikkel Thorup. 用于估计任意子集和的优先采样。*A*CM期刊*(JA*CM*)*，54*(6)*:32，2007年。

[2] Hossein Jowhari, Mert Saglam, and G´abor Tardos. 关于lp采样器、流中查找重复项和相关问题的紧密界限。*CoRR*，abs/1012.4889，2010年。

[3] Mario Szegedy. dlt优先采样基本上是最优的。在计算理论的第三十八届*ACM*研讨会上，页码为150-158。*ACM*，2006年。

CS 598CSC: 大数据算法	讲座日期：2014年9月25日
讲师：Chandra Chekuri	记录员：Chandra Chekuri

假设我们有一个流 a_1，a_2，...，a_n来自一个有序的宇宙。 为了简单起见，我们假设它们是实数，而且它们是不同的（为了简单起见）。 我们希望找到第 k个排名的元素，其中$1 \leq k \leq n$。 特别是，我们可能对中位数感兴趣。 我们将讨论这些问题的精确和近似版本。 寻找排名 k元素的另一种术语是分位数。 给定一个数字 ϕ，其中$0< \phi \leq 1$，我们希望返回一个排名为 ϕn的元素。 这种归一化使我们能够讨论 ϵ-近似分位数，其中 $\epsilon \in [0,1)$。一个 ϵ-近似分位数是一个排名至少为$(\phi - \epsilon)n$且最多为$(\phi + \epsilon)n$的元素。换句话说，我们允许一个加法误差为 ϵn。关于分位数和分位数查询有大量的文献。实际上，Munro和Paterson [5]的一篇最早的"流式处理"论文描述了一种使用$\tilde{O}(n^{1/p})$空间的 p次算法进行选择。他们还考虑了"随机顺序"流式处理设置，这在近年来变得非常流行。

这些讲座的材料主要来自Greenwald和Khanna的优秀章节/调查[1]。 我们主要参考那一章，并在这里描述我们在讲座中涵盖的大纲。 我们将省略证明或给出简略的论证，并参考读者[1]。

1 近似分位数和摘要

假设我们想要能够对有序集合S中的元素进行 ϵ-近似分位数查询。 很容易看出，我们可以简单地选择排名为 $i|\ S|/k$的元素，其中$1 \leq i \leq k \simeq 1/\epsilon$，并将它们作为摘要存储，并使用摘要回答任何分位数查询，具有 ϵn的加法误差。 然而，要获得这些元素，我们首先需要进行选择，如果我们只想要一个简明的摘要，我们可以在离线设置中进行。 我们将在流式设置中解决一个摘要的问题。 接下来，我们将以"字"为单位计算空间使用量。 因此，前面离线摘要的空间使用量为$\Theta(1/\epsilon)$。

我们将看到两个算法。 第一个算法将使用$O(\frac{1}{\epsilon}\log^2(\epsilon n))$的空间创建一个近似总结[4]，受到Munro和Paterson的工作的启发。Greenwald和Khanna [2]提供了一个使用$O(\frac{1}{\epsilon}\log(\epsilon n))$空间的不同总结。

根据[1]，我们将把一个分位数总结 Q看作是存储一组元素 $\{q_1, q_2, \ldots, q_\ell\}$以及每个$q_i$的区间$[\mathbf{rmin}_Q(q_i), \mathbf{rmax}_Q(q_i)]$； $\mathbf{rmin}_Q(q_i)$是 q_i在 S中的秩的下界， $\mathbf{rmax}_Q(q_i)$是 q_i在 S中的秩的上界。 方便起见，我们假设 $q_1 < q_2 < \ldots < q_\ell$，并且 q_1是 S中的最小元素， q_ℓ是 S中的最大元素。 为了简化表示，我们将简单地使用 Q来指代分位数总结，以及有序集合 $\{q_1, q_2, \ldots, q_\ell\}$。

我们的第一个问题是询问一个分位数摘要 Q是否可以用来给出 ϵ-近似的分位数查询。 以下是直观的，并且值得自己证明。

引理1 假设 Q是一个对于 S 的分位数摘要，使得对于 $1 \leq i < \ell$，$_Q(q_{i+1}) - _Q(q_i) \leq 2\epsilon|\ S|$。 那么 Q可以用于对S进行 ϵ-近似的分位数查询。

在接下来的内容中，当我们说 Q是一个 ϵ-近似的分位数摘要时，我们将隐含地使用前面引理中的条件。

给定一个分位数摘要 Q'对于一个多重集合 S'和一个分位数摘要 Q''对于一个多重集合 S''，我们希望将 Q'和 Q''组合成一个分位数摘要 Q对于 $S = S' \cup S''$；在这里，我们将 S视为一个多重集合。 当然，我们希望保持结果摘要的近似程度与 Q'和 Q''相似。 这里有一个引理，表明我们确实可以轻松地组合。

引理**2** 设 Q'为多重集合 S'的一个 ϵ'-近似分位数摘要， Q''为多重集合 S''的一个 ϵ''-近似分位数摘要。 那么 $Q = Q' \cup Q''$是一个 ϵ-近似分位数摘要对于 $S = S' \cup S''$其中 $\epsilon \leq \frac{\epsilon' n' + \epsilon'' n''}{n' + n''} \leq \max\{\epsilon', \epsilon''\}$其中 $n' = |S'|$且 $n'' = |S''|$。

我们不会证明正确性，但会描述如何从 Q'和 Q''构建区间。 假设 $Q' = \{x_1, x_2, \ldots, x_a\}$且 $Q'' = \{y_1, y_2, \ldots, y_b\}$。 考虑某个$z_i \in Q$并假设 $z_i = x_r$，其中$1 \leq r \leq_a$。 设 y_s是 Q''中小于 x_r的最大元素， y_t是 Q''中大于 x_r的最小元素。 我们将忽略 y_s，y_t不被定义的情况。我们设定

$$\mathbf{rmin}_Q(z_i) = \mathbf{rmin}_{Q'}(x_r) + \mathbf{rmin}_{Q''}(y_s)$$

和

$$\mathbf{rmax}_Q(z_i) = \mathbf{rmax}_{Q'}(x_r) + \mathbf{rmax}_{Q''}(y_t) - 1.$$

很容易证明上述为有效区间。 然后可以证明，在这些设置下，对于 $1 \leq i < a + b$ ，以下成立：

$$\mathbf{rmax}_Q(z_{i+1}) - \mathbf{rmin}_Q(z_i) \leq 2\epsilon(n' + n'').$$

我们将上述操作称为 $\mathrm{COMBINE}(Q', Q'')$。 以下很容易看出。

引理**3** 令 $Q_1, \ldots, Q_h$ 为 ϵ-近似分位数摘要，分别对应于 $S_1, S_2, \ldots, S_h$。 然后 Q_1，Q_2，. . . ，Q_h可以以任意顺序组合以获得一个 ϵ-近似分位数摘要 $Q = Q_1 \cup \ldots \cup Q_h$对于 $S_1 \cup \ldots \cup S_h$.

接下来我们讨论在分位数摘要 Q上的 $\mathrm{PRUNE}(Q, k)$ 操作，该操作将Q的大小减小到 $k+1$，同时在近似质量上损失一点。

引理**4** 设 Q'是 S的一个 ϵ-近似分位数摘要。 给定一个整数参数 k 存在一个分位数摘要 $Q \subseteq Q'$对于 S，使得 $|Q| \leq k+1$且它是 $(\epsilon + \frac{1}{2k})$-近似的。

我们概述证明过程。 我们只需查询 Q'的排名 1，$|S|/\mathrm{k}$，$2|S|/\mathrm{k}$，. . . ，$|S|$并选择这些元素放入 Q。 我们保留它们在 Q'中的 **rmin**和 **rmax**值。

$$\mathbf{rmax}_Q(q_{i+1}) - \mathbf{rmin}_Q(q_i) \leq i|S|/k + \epsilon|S| - ((i-1)|S|/k - \epsilon|S|) \leq |S|/k + 2\epsilon|S|.$$

1.1 一个 $O(\frac{1}{\epsilon}\log^2(\epsilon n)$空间算法

这个想法受到了Munro-Paterson算法的启发，并在Manku等人的论文中进行了抽象。尽管它可以在流计算中实现，但我们将以离线方式描述这个想法。 我们将使用几个具有 k个元素的分位数摘要，其中 k是某个参数，比如$\ell$of them. 每个大小为 k的摘要将被称为一个缓冲区。 随着流中的更多元素到达，我们需要重复使用这些缓冲区；缓冲区将被合并和修剪，换句话说，“折叠”成相同大小的缓冲区 k。 修剪会引入误差。

为了简单起见，假设n/k是$_2$的幂。 考虑一个完全二叉树，其中有n/k个叶子节点，每个叶子节点对应于流中的k个连续元素。 想象一下为了获得这k个元素的0误差分位数摘要而分配一个大小为k的缓冲区；从技术上讲，我们需要$k+_1$个元素，但为了清晰起见，我们将忽略这个小的附加问题。 现在，树的每个内部节点对应于流的一部分元素。 想象一下为了维护子树中流元素的近似分位数摘要而为每个内部节点分配一个大小为k的缓冲区。 为了在节点v处获得一个摘要，我们将其两个子节点v_1和v_2的摘要合并，并将其修剪为大小k，引入额外的$_1/(_2k)$的误差。 大小为k的根节点的分位数摘要将是我们对流输出的最终摘要。

我们的第一个观察是，实际上我们可以用 $\ell = O(h)$缓冲区来实现基于树的方案，其中 h是树的高度。 注意 $h \simeq \log(n/k)$。 我们只需要 $O(h)$缓冲区的原因是，如果需要为流中的下一个 k个元素创建一个新的缓冲区，我们可以合并两个对应于内部节点的子节点的缓冲区 - 因此，任何时候我们只需要维护树的每一层一个缓冲区（加上一个临时缓冲区来执行合并操作）。

考虑叶子节点上的分位数摘要。 它们的误差为0，因为我们在缓冲区中存储了所有元素。 然而，在每个层级上，误差增加了$1/(2k)$。 因此，根节点的摘要的误差为 $h/(2k)$。 因此，为了获得一个 ϵ-近似的分位数摘要，我们需要 $h/(2k) \leq \epsilon$。
并且 $h = \log(n/k)$. 可以看出，为了使其工作，选择$k > _2\frac{1}{\epsilon}\log(2\epsilon n)$就足够了。 —
总空间使用量为$\Theta(hk)$，且 $h = \log(n/k)$，因此空间使用量为 $O(\frac{1}{\epsilon}\log^2(\epsilon n))$。 -
可以选择 d叉树而不是二叉树，并进行一些优化以改善
常数，但是这种高级方案对 ϵ的渐近依赖并没有改善。

1.2 一个 $O(\frac{1}{\epsilon}\log(\epsilon n)$空间算法

我们现在简要介绍一下Greenwald-Khanna算法，该算法可以获得改进的空间界限。
GK算法将分位数摘要作为一个集合，其中包含 s个元组 $t_0, t_2, \ldots$，其中每个元组 t_i是一个三元组 (v_i, g_i, Δ_i)：(i) 值 v_i是有序集合 S的一个元素 (ii) 值 g_i等于 $\mathbf{rmin}_{\mathrm{GK}}(v_i)_- \mathbf{rmin}_{\mathrm{GK}}(v_i - _1)$ (对于 $i = 0$， $g_i = 0$) 和 (iii) 值 Δ_i等于 $\mathbf{rmax}_{\mathrm{GK}}(v_i)_- \mathbf{rmin}_{\mathrm{GK}}(v_i)$。 元素 $v_0, v_1, \ldots$， $\mathrm{v}_s - _1$按升序排列，而且 v_0是 S中的最小元素， v_{s-1}是 S中的最大元素。注意

$n = \sum$总结还存储了 n迄今为止看到的元素数量。通过这种设置，我们
注意到 $\mathbf{rmin}_{\mathrm{GK}}(v_i) = \sum_{j \leq i} g_j$ 和 $\mathbf{rmax}_{\mathrm{GK}}(v_i) = \Delta_i + \sum_{j \leq i} g_j$.

引理5 假设 Q 是一个GK分位数摘要，用于一个集合 $|S|$ 使得 $\max_i(g_i + \Delta_i) \leq 2\epsilon|S|$
那么它可以用来回答带有 ϵn加法误差的分位数查询。

查询可以如下回答。 给定排名 r，找到 $i$such that $r - \mathbf{rmin}_{\mathrm{GK}}(v_i) \leq \epsilon n$ and
$\mathbf{rmax}_{\mathrm{GK}}(v_i) - r \leq \epsilon n$and输出 v_i；这里 n是 S的当前大小。
通过两个操作来更新分位数摘要。 当一个新元素 v到达时，它被插入到摘要中。 为了保持摘要在所需的空间范围内，连续的元素被合并来压缩分位数摘要。

现在我们描述一下插入操作，它接受一个分位数摘要 Q并插入一个新元素 v。 首先，我们在 Q中搜索元素，找到一个 i使得 $v_i < v < v_{i+1}$；当 v是最小元素或最大元素时，处理起来很容易。 添加一个新的元组$t = (v, 1, \Delta)$，其中 $\Delta = \lfloor 2\epsilon n \rfloor - 1$ 成为新的（$i+1$）个元组。

请注意这里是流的当前大小。 很容易看出，如果摘要 Q之前

满足引理5中的条件，则在插入元组之后，摘要 Q仍然满足条件（注意 n增加了1）。 我们注意到前$1/(2\epsilon)$个元素被插入到摘要中，其中$\Delta_i = 0$。

压缩是主要的组成部分。 为了理解操作，有助于定义元组的容量。 请注意，当 v到达 $t = (v, 1, \Delta)$被插入时，其中$\Delta \simeq 2\epsilon n'$，其中 n'是 v到达的时间。 在时间$n > n'$时，元组 t_i的容量定义为$2\epsilon n - \Delta_i$。 随着 n的增长，元组的容量增加，因为我们允许有更多的误差。

我们可以在时间n将元组$t_{i'}, t_{i'+1}, \ldots, t_i$合并为$t_{i+1}$（这意味着我们消除了$t_{i'}, \ldots, t_i$），同时确保所需的精度，如果$\sum_{j=i'}^{i+1} g_j + \Delta_{i+1} \leq 2\epsilon n$；$g_{i+1}$被更新为$\sum_{j=i'}^{i+1} g_j$，并且$\Delta_{i+1}$不变。请注意，这意味着一旦插入元组，其$\Delta$值就不会改变。

请注意，插入和合并操作保持了摘要的正确性。 为了获得所需的空间限制，合并/压缩必须非常小心地进行。 我们不会详细介绍，但提到一个关键思想是在几何递增的间隔内跟踪元组的容量，并确保摘要每个间隔只保留少量的元组。

2 精确选择

我们现在将描述一个使用$\tilde{O}(n^{1/p})$空间的p次确定性算法，在流中选择排名为k的元素；这里$p \geq 2$。很容易证明对于$p = 1$，任何确定性算法都需要$\Omega(n)$空间；在关于位与字的精确模型的论证中，需要小心一点，但是近似线性的下界很容易得到。 Munro和Paterson描述了使用$\tilde{O}(n^{1/p})$空间的算法，需要p次遍历。 我们不会描述他们的精确算法，而是使用近似分位数的分析。

我们将证明，给定空间 s和一个流 n的项，该问题可以在一次遍历中有效地减少到选择 $O(n \log^2 n/s)$个项。

假设我们可以做到上述。 经过 i次遍历后，问题被减少到 $n^{p-i}\overline{p}(\log n)^{\frac{2i}{p}}$个元素。 设置 $i = p - 1$，我们可以看到剩下的元素数量为 $O(s)$。 因此，所有这些元素都可以被存储，选择可以离线进行。

现在我们描述如何使用一次遍历将考虑的元素的有效大小减少到 $O(n \log^2 n/s)$。 思路是我们将能够从流中选择两个元素 a_1，b_1，使得 $a_1 < b_1$，并且第 k个排名的元素保证在区间$[a_1, b_1]$内。

此外，我们还保证流中a和b之间的元素数量为$O(n \log^2 n/s)$。 a_1和 b_1是第一次通过后的左右过滤器。. 最初， $a_0 = -\infty$和 $b_0 = \infty$。

经过i次通过后，我们将得到过滤器 a i和 b_i。 请注意，在$(i + 1)$次通过期间，我们可以计算出 a_i和b_i的精确排名。

我们如何找到 a_1和 b_1？ 我们已经看到如何使用$O(\frac{1}{\epsilon} \log^2 n)$的空间获得一个$\epsilon$-近似摘要。 因此，如果我们有空间 s，我们可以设置 $\epsilon' = \log^2 n/s$。让 $Q = \{q_1, q_2, \ldots, q_\ell\}$为流的 ϵ'-近似分位数摘要。 我们查询 Q以获得 $r_1 = k - \epsilon' n - 1$和 $r_2 = k + \epsilon' n + 1$，并获得 a_1和b_1作为查询的答案（在这里我们忽略了 $r1 < 0$或 $r_2 > n$的特殊情况）。

然后，根据 Q的ϵ-近似保证，我们可以得出排名 k的元素位于区间 $[a_1, b_1]$，而且在这个范围内最多有 $O(\epsilon' n)$个元素。

计算 $p = 2$时，对代数进行推导是有用的，这表明中位数可以在 $O(\sqrt{n} \log^2 n)$的空间内计算出来。

2.1 随机顺序流

Munro和Paterson在他们的论文中也考虑了随机顺序流模型。 在这里，我们假设流是有序集合的随机排列。使用一个不同的模型更加方便，其中第 i个元素是从区间[0,1]中独立抽取的实数。 我们可以问是否可以利用随机性。确实可以。他们证明了用 $O(\sqrt{n})$的空间，可以高概率地找到中位数。 更一般地，他们证明了在 p次通过中，可以用空间 $O(n^{1/(2p)})$找到中位数。 尽管这个空间限制比对抗性流更好，但如果我们只有多对数空间，仍然需要$\Omega(\log n)$次通过，与对抗性设置相同。 Guha和McGregor [3]证明了实际上只需要 $O(\log\log n)$次通过（高概率下）。

在这里，我们描述了Munro-Paterson算法；也可以参考`http://polylogblog.wordpress.com/2009/08/30/bite-sized-streams-exact-median-of-a-random-order-stream/`。

该算法在迄今为止的流中维护了一个连续排名的元素集合 S of s。它维护了两个计数器 ℓ，用于记录小于最小值 S（即 S中的最小元素）的元素数量和大于最大值 S的元素数量 h。它试图将 $h-\ell$尽可能接近0以“捕获”中位数。

```
MunroPaterson(s):

n ← 0
S ← ∅
ℓ, h ← 0
当（流不为空）时
  n ← n + 1
  a是新元素
  如果（a < min S）则 ℓ ← ℓ+ 1
  否则如果（a > max S）则 h ← h+ 1
  否则将 a添加到 S
  如果（|S| = s+ 1)
      如果 (h < ℓ) 则从 S中丢弃最大的 S并将 h ← h+ 1
      否则从 S中丢弃最小的 S并将ℓ ← ℓ+ 1
结束循环
如果 1 ≤ (n+ 1)/2 − ℓ ≤ s则
  返回 S中第(n+ 1)/2 − ℓ小的元素作为中位数
否则返回失败。
```

为了分析算法，我们考虑随机变量 $d = h - \ell$，它从0开始。 在前 s次迭代中，我们只是简单地填满 S直到容量满，并且 $h-\ell$保持为0。 之后，在每一步中， d要么增加1，要么减少1。 考虑当$i > s$时的迭代开始。 在迭代 i开始之前，已经看到的元素总数是 $i-1 = \ell + h + s$。 在迭代 i中，由于排列是随机的， a_i大于最大的 S的概率恰好是 $(h+1)/(h+s+1+\ell)$。 而 a_i小于最小的 S的概率恰好是 $(\ell+1)/(h+s+1+\ell)$，因此以概率 $(s-1)/(h+s+1+\ell)$， a_i将被添加到 S中。

请注意，只有当流的结尾处的 $|d|$大于 s时，算法才会失败。 成功的充分条件是，在整个算法过程中， $|d| \le s$始终成立。 设 $p_{d,i}$为在条件$0 < |d| < s$下， $|d|$增加1的概率。 那么我们可以看到， $p_{d,i} \le 1/2$。 因此，这个过程可以看作是在一条线上的随机游走，通过一些分析可以得出，如果我们选择 $s = \Omega(\sqrt{n})$，那么在很大的概率下， $|d| < s$始终成立。 因此，当流是随机顺序时，$\Omega(\sqrt{n})$的空间足以在很大的概率下找到中位数。

与**CountMin**草图和删除的关联：请注意，当我们讨论频率矩时，我们假设元素是从一个$[n]$中抽取的，其中 n是事先已知的，而在这里，我们对元素除了它们来自一个有序的宇宙之外没有做任何假设（对于之前使用的 m的长度表示法的混淆表示道歉）。 如果我们事先知道元素的范围，并且它相对于流的长度很小，那么**CountMin**和相关技术更适合，并且提供处理删除的能力。 GK摘要也可以处理一些删除操作。 更多细节请参阅[1]。

下界：对于中位数选择，M**unro**和Pat**erson**在计算模型的限制下，证明了空间的下界为$\Omega(n^{1/p})$，需要 $p$pa**sses**。 G**u**ha和Mc**G**re**g**o**r**在没有任何限制的情况下，证明了下界为$\Omega(n^{1/p}/p^6)$ 位。 对于随机顺序的流，使用$O(\log\log n)$ pa**sses**和p**o**ly**log**(n)空间可以高概率地进行精确选择[3]。 此外，$\Omega(\mathbf{log}\log\mathbf{log}\,n)$pa**sses**确实是必要的；参见[3]中的引用和讨论。

参考文献注释：更多信息请参考参考文献。

参考文献

[1] Michael Greenwald和Sanjeev Khanna。流上的分位数和等深直方图。可在`http://www.cis.upenn.edu/~mbgreen/papers/chapter.pdf`上获取。 将作为即将出版的数据流管理书籍的一章出现。

[2] Michael Greenwald 和 Sanjeev Khanna. 空间高效的在线计算分位数摘要。 在*ACM SIGMOD Record*, 卷 30, 页码 58–66. ACM, 2001年。

[3] Sudipto Guha 和 Andrew McGregor. 流次序和次序统计：在随机次序流中的分位数估计。 *SIAM Journal on Computing*, 38(5):2044–2059, 2009年。

[4] Gurmeet Singh Manku, Sridhar Rajagopalan 和 Bruce G Lindsay. 一次通过有限内存近似中位数和其他分位数。 在*ACM SIGMOD Record*, 卷 27, 页码 426–435. ACM, 1998年。

[5] J Ian Munro 和 Mike S Paterson. 有限存储的选择和排序。 理论计算机科学, 12(3):315–323, 1980年。