大数据计算基础

BIG DATA COMPUTING

刘显敏 海量数据
liuxianmin@hit.edu.cn

大数据算法

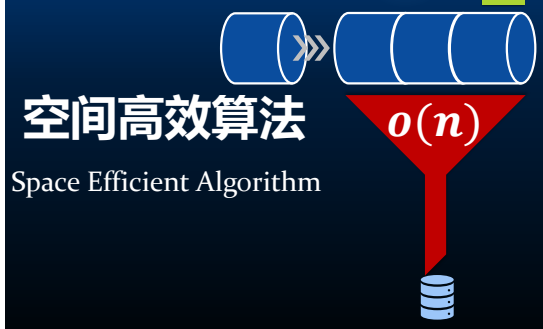
2025年秋

2

空间高效算法

Space Efficient Algorithm

$o(n)$



流模型/One-Pass计算模型

3

- 输入是数据流的方式，可由$\langle a_1, a_2, \ldots, a_n\rangle$表示
  - 例1：每个数据是$[1, m]$之间的整数
  - 例2：每个数据是一条边(u, v)
- 允许空间代价 sublinear $o(n)$ polylog $O(\log^c n)$
- 流算法的分析维度
  - 空间代价
  - 时间代价 worst-case time for each data
  - 全体时间代价 相当于 amortized-case time
  - 输出结果质量
  - 算法成功的概率 如果是随机算法

路由器监测
数据库查询

multi-pass
semi-streaming

问题1：近似计数

4

流模型（Streaming Model）

- 给定数据流 $\sigma = \langle a_1, a_2, \ldots, a_n\rangle$，其中$a_i \in [m]$
- 对应的频次向量为

$$\boldsymbol{f} = (f_1, \ldots, f_m), s.t. \sum_{1\le i\le m} f_i = n.$$

定义 [The Counting Problem]
给定数据流σ以及a_i，计算f_i。

问题1：近似计数

5

为了计算 f_i，假设 a_i 共出现 n 次

- 需要 $O(\log n)$ 位的空间代价存储 n
- 如果 n 非常大， $O(\log n)$ 也许不可接受
- 是否有可能占用更少的空间代价？
  - 当然不可能，除非我们放宽要求
- 我们将设计一个占用$O(\log\log n)$位空间的近似算法

定义 [The Approximate Counting Problem]
给定数据流σ以及a_i，计算$\widehat{f_i}$，满足

$$\mathbf{Pr}\left[\left|\widehat{f_i} - f_i\right| > \epsilon f_i\right] < \delta$$

$\epsilon = 1/3$
$\delta = 1\%$

问题1：近似计数

6

Morris算法

/** 维护一个计数器X **/
1. $X \leftarrow 0$;
2. **for** 每一个元素a_i **do**
   以概率$\frac{1}{2^X}$更新$X \leftarrow X + 1$;
3. **return** $\widehat{f_i} = 2^X - 1$

离散型随机变量X
期望

$$\mathbf{E}[X] = \sum_i i \cdot \mathbf{Pr}(X = i)$$

方差

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

Morris算法近似性能分析

- 期望 $\mathbf{E}[2^X - 1] = f_i$
- 方差 $\mathbf{var}[2^X - 1] = f_i(f_i - 1)/2 \le f_i^2/2 = O(f_i^2)$
- 利用切比雪夫不等式

问题1：近似计数 7

- 两个有用的概率不等式

马尔可夫(Markov)不等式

对任意非负随机变量X以及常数$a > 0$，有

$$\mathbf{Pr}(X \geq a) \leq \frac{\mathbf{E}[X]}{a}$$

切比雪夫(Chebyshev)不等式

$$\mathbf{Pr}(|X - \mathbf{E}[X]| \geq a) \leq \frac{\mathbf{var}[X]}{a^2}$$

问题1：近似计数 8

- 切比雪夫：$\mathbf{Pr}(|Z - \mathbf{E}[Z]| \geq a) \leq \frac{\mathbf{var}[Z]}{a^2}$
  - 令$a = \sqrt{\frac{\mathbf{var}[Z]}{c}}$，其中$c$是任意常数
  - 则有，$\mathbf{Pr}\left(|Z - \mathbf{E}[Z]| \geq \sqrt{\frac{\mathbf{var}[Z]}{c}}\right) \leq c$
  - 即，$Z = \mathbb{E}[Z] \pm \sqrt{\mathbf{var}[Z]/c}$ 至少以1-c概率成立
- $Z = 2^X - 1$，至少以$1 - c$概率，Morris算法

$$\hat{f}_i = 2^X - 1 = \mathbb{E}[2^X - 1] \pm \sqrt{\frac{\mathbf{var}[2^X - 1]}{c}}$$

问题1：近似计数 9

- 至少以$1 - c$概率，Morris算法

$$\hat{f}_i = 2^X - 1 = \mathbb{E}[2^X - 1] \pm \sqrt{\frac{\mathbf{var}[2^X - 1]}{c}}$$

$\mathbf{E}[2^X - 1] = f_i$

$\mathbf{var}[2^X - 1] \leq f_i^2/2 = O(f_i^2)$

$$\hat{f}_i = f_i \pm \sqrt{\frac{1}{2c}} f_i = \left(1 \pm \sqrt{\frac{1}{2c}}\right) f_i$$

即 $\hat{f}_i = f_i \pm O(f_i)$

问题1：近似计数 10

Morris算法的**空间代价分析**

定理 [Jensen's inequality]

令Z为随机变量，$\mathbb{E}[Z] < \infty$，如果f是凸函数，那么有$f(\mathbb{E}[Z]) \leq \mathbb{E}[f(Z)]$

- 令$f(y) = 2^y$，有$2^{\mathbb{E}[X]} \leq \mathbb{E}[2^X] = f_i + 1$

$$\mathbb{E}[X] \leq \log(f_i + 1)$$

$$\Rightarrow \mathbb{E}[\text{space}(X)] \approx \text{loglog} f_i = O(\text{loglog } n)$$

问题1：近似计数 12

如何获得一个$1 + \epsilon$近似算法

Morris算法： $\hat{f}_i = f_i \pm \sqrt{\frac{1}{2c}} f_i = \left(1 \pm \sqrt{\frac{1}{2c}}\right) f_i$

2. **return** $\frac{1}{k}\sum_{i=1}^{k}(2^{X_i} - 1)$;

Morris+算法的估计结果分析

- 期望$\mathbf{E}[Y] = \mathbf{E}\left[\frac{1}{k}\sum_{i=1}^{k}(2^{X_i} - 1)\right] = f_i$
- 方差$\mathbf{var}[Y] = \frac{f_i(f_i - 1)}{2k} \leq \frac{f_i^2}{2k} = O\left(\frac{f_i^2}{k}\right)$
- 利用切比雪夫不等式

问题1：近似计数 13

定理 [Morris+]至少$1 - c$概率，Morris+算法满足

$$Y = f_i \pm \frac{1}{\sqrt{2kc}} f_i$$

- 令$k = \frac{1}{2c\epsilon^2} = O\left(\frac{1}{\epsilon^2}\right)$，Morris+是$(1 \pm \epsilon)$近似算法
- 令$k = \frac{1}{2\delta\epsilon^2} = O\left(\frac{1}{\epsilon^2}\frac{1}{\delta}\right)$，可将概率改进为$1 - \delta$
  - 空间代价为$O\left(\frac{\text{loglog } n}{\epsilon^2\delta}\right)$

问题1：近似计数 14

- 以$1-\delta$概率$(1\pm\epsilon)$-近似, Morris+需$k=O(\frac{1}{\epsilon^2}\frac{1}{\delta})$
- 利用Morris+算法$\mathcal{M}$：以概率$1-c \geq 0.9$得到$1\pm\epsilon$近似，设计一个$1-\delta$概率$(1\pm\epsilon)$-近似算法
- 空间代价为$O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log\log\right)$

Morris++算法

*/** Median Trick **/*

1.同时独立执行 $t=O(\log(1/\delta))$个$\mathcal{M}$算法；

2.取 t 次估计结果的中间值(median)返回；

算法分析
(1)计算期望
(2)利用切尔诺夫不等式

问题1：近似计数 15

定理 [切尔诺夫不等式, Chernoff/Hoeffding Bound]

令$X_1, X_2, \ldots, X_m$是独立、取值{0, 1}的随机变量，变量和的期望为$\mu=\mathbb{E}[\sum_i X_i]$，令$\epsilon\in[0,1]$，则有

$$\Pr\left[\left|\sum_i X_i-\mu\right|>\epsilon\mu\right]\leq 2e^{-\epsilon^2\mu/3}$$

问题1：近似计数 16

算法分析

- $X_i=1\Leftrightarrow$第i个Morris+算法$\mathcal{M}$输出结果符合要求

$[\sum_i^t$ $9t$

若要$\Pr[\sum_i X_i<0.5t]\leq e^{-c't}<\delta$
只需$t=O\left(\log\left(\frac{1}{\delta}\right)\right)$

- 易知，$\ldots$法输出结果$\ldots$

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

$\Rightarrow$ 空间代价：$O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log\log n\right)$

$\leq 2e^{\ldots}$ $\leq e^{\ldots}$

问题3：固定大小采样 17

- 随机采样是一个非常重要的工具
- 随机采样：从大小为n的集合中均匀选择s个样本
  - 一个想法：遍历数据，以 $\frac{s}{n}$ 的概率选择每个数据
  - 缺点：事先需要知道n，无法保证样本集合大小

水库抽样算法–样本大小为1

1. n ← 0;
2. **for** 每一个元素 x **do**
3. n ← n + 1;
4. 以 $\frac{1}{n}$ 概率令S ← x;
5. **return** S;

问题3：固定大小采样 18

定理[水库抽样算法的正确性]

令n为当前数据流的长度，水库抽样算法的输出是均匀的，即对任意的$i\in[1,n]$，$\Pr[S=x_i]=\frac{1}{n}$。

$$\Pr[S=x_i]=\frac{1}{i}\prod_{n\geq j>i}\left(\frac{j-1}{j}\right)=\frac{1}{n}$$

- 获取大小为s的均匀采样（有放回情况）
  - 将样本大小为1的算法同时独立执行s份

问题3：固定大小采样 19

- 获取大小为s的均匀采样（无放回情况）

水库抽样算法

1. n ← 0;
2. 用流数据的前 s 个元素初始化 A[1, . . . ,s]，n ← s;
3. **for** 每一个元素 x **do**
4. n ← n + 1;
5. 令r为[1, n]内的随机整数;
6. **if** r ≤ s **then**
7. A[r] ← x;

- 证明：A是从数据流中以无放回形式均匀抽取的样本

问题3：固定大小采样 20

- 扩展到weighted的情况
- 当s = 1时，是以$\mathbf{Pr}[S = x_i] = \frac{w_i}{\sum w_i}$概率选取数据
  - 水库抽样算法可以很容易扩展到这个情况

带权重水库抽样算法–样本大小为1

1. W ← 0;
2. **for** 每一个元素 x **do**
3. W ← W + w_x;
4. 以 $\frac{w_x}{W}$ 概率令S ← x;
5. **return** S;

- 样本大小为s的有放回采样：同时独立运行算法s次

问题3：固定大小采样 21

- 有权重、无放回、样本大小为s
- 定义可以通过观察如下概念式算法来理解
  - 当前数据流总权重W，以$\mathbf{Pr}[x_i] = \frac{w_i}{W}$选一个数据
  - 重复上述步骤s次，注意每次W不同

带权重水库抽样算法–无放回抽取s个样本

1. **for** 每一个元素 xi **do**
2. r_i ← (0, 1)间的均匀随机数;
3. $w_i' = r_i^{1/w_i}$;
4. S[1, . . . ,s]为最大的s个w'对应的s个数据;
5. **return** S;

问题4：Bloom Filter 22

- Bloom Filter（布隆过滤器）由Bloom于1970年提出
  - Burton H. Bloom. **Space/time trade-offs in hash coding with allowable errors**. *Commun. ACM*, 13(7), 1970, 422–426.
- 例1：垃圾邮件检查
  - 手头有一个垃圾邮件来源地址列表
- 例2：广告推荐、订阅-发布系统
  - 关键词列表
- 简单做法：遍历一遍名单或者列表
- 大数据场景下，名单太长怎么办

问题4：Bloom Filter 23

定义[Membership Problem]令 U 是整数集合 {1, . . . , |U|}，S 是 U 的一个大小为 n 的子集合，预处理 S，给定整数 q ∈ U，判断是否 q ∈ S。

- U 中整数可以用 w = log |U| 位表示
- 哈希表，查询期望时间是 O(1)
  - 如果要求精确答案，那么至少用$\log\binom{|U|}{n}$位，当$|U| \gg n$，即 Ω(nw)
  - 是否可以在 o(nw) 代价解决?

问题4：Bloom Filter 24

- 放松要求
  - √ 错误判断q ∈ S（false positives）
  - × 错误判断q ∉ S（false negatives）

定义[Approximate Membership Problem]给定整数 q ∈ U，设计算法 ① q ∈ S ⇒输出 ‘yes’ ，② q ∉ S⇒以至少 1 – δ 概率输出 ‘no’ 。

问题4：Bloom Filter 25

定义[Ideal Hash Function] 哈希函数 h : U → [m]是理想的，若对每个 k ∈ U，h(k) 均匀独立地从[1, m]间取值。

近似哈希的方法

1. 令$\mathcal{H}$是独立理想哈希函数族：|U| → [m]，m = $\frac{n}{\delta}$;
2. 随机选取 h ∈ $\mathcal{H}$ ，并维护数组A[m];
3. **for** i ∈ S **do** A[h(i)] = 1;
4. 给定查询 q，**return** ‘yes’ 当且仅当A[h(i)] = 1;

- $q \in S$ ⇒输出 ‘yes’
- $q \notin S \Rightarrow \sum_{j \in S} \mathbf{Pr}[h(q) = h(j)] \leq \frac{n}{m} = \delta$

问题4：Bloom Filter

26

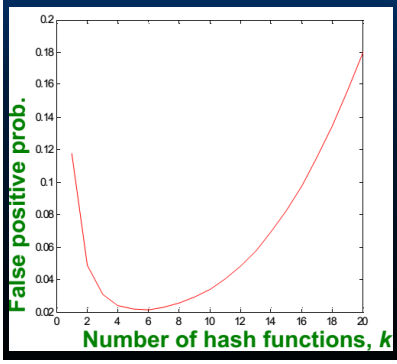
Bloom Filter方法

1. 令$\mathcal{H}$是独立理想哈希函数族：|U| → [m];
2. 随机选取$h_1, \ldots, h_k \in \mathcal{H}$，并维护数组 A[m];
3. **for** i ∈ S **do**
4. **for** j ∈ [1, k] **do**
5. A[h_j(i)] = 1;
6. 对查询 q，**return** ‘yes’ ⇔ ∀j ∈ [k], A[h_j(q)] = 1;

- 代价$m = O(n \log \frac{1}{\delta}) \Rightarrow O(n \log \frac{1}{\delta})$空间代价

问题4：Bloom Filter

27



性质总结

- 需要被过滤掉的一定被过滤
- 适合做预处理
- 适合硬件实现
- 适合并行

问题4补充：哈希（hashing）

28

- 查找技术简述

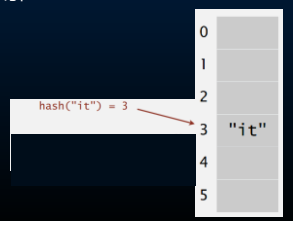
implementation	guarantee			average case			ordered ops?
	search	insert	delete	search hit	insert	delete	
sequential search (unordered list)	N	N	N	½ N	N	½ N	
binary search (ordered array)	lg N	N	N	lg N	½ N	½ N	✔
BST	N	N	N	1.39 lg N	1.39 lg N	$\sqrt{N}$	✔
red-black BST	2 lg N	2 lg N	2 lg N	1.0 lg N	1.0 lg N	1.0 lg N	✔

- 能做的再好一些吗?

问题4补充：哈希（hashing）

29

- 哈希的想法（hashing）
  - 将数据存储在key-indexed数组中，即以“key”值来确定数组索引下标(index)位置
- 哈希函数
  - 输入key，输出数组的位置索引
- 需要解决的问题
  - ①如何设计哈希函数
  - ②如何处理“冲突”



问题4补充：哈希（hashing）

30

①如何设计哈希函数

- 理想目标：将键值(key)尽可能“**打散**”，使其**均匀分布**到所有可能索引下标的范围内
  - 要易于计算
  - 每个key被映射到不同索引下标的可能性是相同的
- 例1：电话号码
  - 不好的设计：取前3位
  - 较好的设计：取后3位
- 例2：身份证号
  - 不好的设计：取前4位
  - 较好的设计：取后4位

要根据数据来设计哈希函数

问题4补充：哈希（hashing）

31

①如何设计哈希函数

- Java语言里的实现

```
public final class Double                                    数值
{
   private final double value;
   ...

   public int hashCode()
   {
      long bits = doubleToLongBits(value);
      return (int) (bits ^ (bits >>> 32));
   }
}
```

问题4补充：哈希（hashing） 32

例子: s= “call”，s.hashCode()如下:

$$99 \cdot 31^3 + 97 \cdot 31^2 + 108 \cdot 31^1 + 108 \cdot 31^0 = 3045982$$

本质：为一个长度为L的字符串计算

$$s[0] \cdot 31^{L-1} + \cdots + s[L-3] \cdot 31^2 + s[L-2] \cdot 31^1 + s[L-1] \cdot 31^0$$

```
public int hashCode()
    for (int i = 0; i < length(); i++)
        hash = s[i] + (31 * hash);
    return hash;
}
```

为什么是31？？

ith character of s

char	Unicode
…	…
'a'	97
'b'	98
'c'	99
…	…

问题4补充：哈希（hashing） 33

①如何设计哈希函数

- 取模(modular)：Java语言里的实现
  - hashCode(): -2^{31}~$2^{31}-1$之间的整数
  - Hash函数需求：0~$M-1$之间的整数(假设数组大小为M)

计算机表示有理数（补码编码）

- 用最高位(bit)表示

会出现负数

2^{-31}时会出现负数

```
private int hash(Key key)
{  return key.hashCode() % M;  }

private int hash(Key key)
{  return Math.abs(key.hashCode()) % M;  }

private int hash(Key key)
{  return (key.hashCode() & 0x7fffffff) % M;  }
```

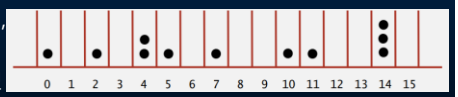
问题4补充：哈希（hashing） 34

①如何设计哈希函数

- 单个哈希函数效果如何?
  - 均匀哈希假设：每一个键值会被等可能地映射到0到M-1的整数

大量使用后，效果如何?

- 球箱模型



- ①大约$\sqrt{\frac{\pi M}{2}}$次使用后，会出现两个球落到一个箱子【生日悖论】
- ②大约$M \ln M$次使用后，没有空箱子【赠券收集问题】
- ③使用M次后，最满箱子的期望球数是$\Theta\left(\frac{\log M}{\log \log M}\right)$【负载均衡】

小说《双城记》中文字的哈希结果

问题4补充：哈希（hashing） 35

②如何处理“冲突”

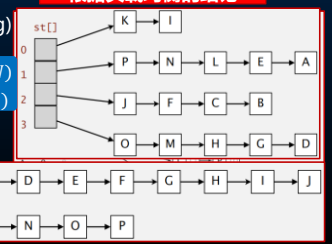
- 冲突的概念：不同键值被哈希(hash)到同一个索引下标
  - 冲突无法避免
  - 冲突分布会比较均匀

根据生日悖论，$\sqrt{M}$个键值就会引发冲突

根据负载均衡的结论

- 链式(separate chaining)
  - 插入操作
  - 查找操作
  - 删除操作
  - Resize+rehash

最坏情况$O(N)$
平均情况$O(1)$



问题4补充：哈希（hashing） 36

②如何处理“冲突”

- 线性探索(linear probing)法
  - 开放定址法的一种，冲突时在数组中找其它空位
  - 如果某个键值对应的哈希索引i处被占用，则依次检测后续位置，$i+1$、$i+2$、…、$M-1$、0、1、…、$i-1$
- hash(K) = 5，查询K? 插入K?

删除是比较麻烦的情况，不能一删了之

- 假设N个键值，M个位置，显然$M \geq N$，分析displacement
- 半满：$N = \frac{M}{2}$，平均displa
- 全满：$N = M$，平均displa

最坏情况$O(N)$
平均情况$O(1)$

最常用的设计策略
$\frac{N}{M} \sim \frac{1}{2}$

问题4补充：哈希（hashing） 37

②如何处理“冲突”

- **Two-probe hashing**：链式的变种；映射到2个位置，插入到较短的链中
- **Double hashing**：线性探索变种；每次不是加1，加一个可变的数
- **Cuckoo hashing**：线性探索变种；映射到2个位置，插入到哪个位置都行

问题5：LSH技术 38

- 局部敏感哈希技术(locality sensitive hashing)
- 应用场景
  - 从很多文档中找到“相似”的文档
  - 从很多网页中找到“相似”的网页
  - 从一堆集合中找到“相似”的集合
  - 从一堆图片中找到“相似”的图片
  - 从一堆向量中找到“相似”的向量

问题5：LSH技术 39

- 举例：相似文档问题
  - 输入：一个文档的集合
  - 输出：相似的文档对(pair)
- 寻找文本内容相似的文档，主要指字面上的相似，而非语义
- 检查相同，可逐字比较；很多应用中不相同，只是大部分相同
- 应用场景
  - 抄袭文档。抄袭者可能会从其它文档中将某些部分的文档据为己有，同时可能对某些词语或者原始文本中的次序进行改变。
  - 镜像页面。重要的web站点会在多个主机建立镜像页面，这些镜像的主要内容相似，但是也包括不同的内容（每个站点都指向其他站点而不指向自己），搜索引擎需过滤掉内容相同的镜像站点
  - 同源新闻稿。一个记者把一个新闻稿件投到多家报刊。每家修改后刊发。Google new能够发现此类稿件，只显示一个版本。

问题5：LSH技术 40

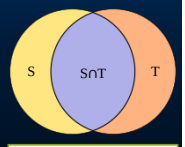
- 举例：协同过滤问题
  - 向用户推荐相似用户所喜欢的那些项
- 应用场景：在线购物
  - 两个用户兴趣相似，如果他们购买的商品集合有较高的Jaccard相似度，20%就很高了。
  - 两个商品相似，如果顾客集合有较高的Jaccard相似度
  - 可能需要一些辅助工作来发现相似。如两个顾客各自购买了大量科幻小说，但这些小说不相同，通过相似度发现和聚类，把这些科幻小说归为一类，从而提高这两个顾客的相似度。
- 应用场景：电影评级
  - NETFLix记录了用户租借电影的情况，还记录了顾客对电影的打分，如果电影被许多相同的用户租借或打分，则这些电影相似
  - 如果用户租借很多相同的电影或者打分类似，则这些用户相似。

问题5：LSH技术 41

- 举例：相似文档问题
  - 输入：一个文档的集合
  - 输出：相似的文档对(pair)
- 抽象为计算问题
  - k-shingles：连续k个单词构造的字符串
  - 文档的k-shingles相似，则文档相似
  - 例如：Text Document = “What is the likely date that the regular classes may resume in Ontario”
  - **2−shingles**: What is, is the, the likely, . . . , in Ontario
  - **3−shingles**: What is the, is the likely, . . . , resume in Ontario
  - 实际应用中，英文文档用9-shingles，邮件用5-shingles

问题5：LSH技术 42

- 举例：相似文档问题
  - 输入：一个文档的集合
  - 输出：相似的文档对(pair)
- 抽象为计算问题（进一步）
  - 文档D →集合S：S是D中所有的k-shingles
  - 雅卡尔相似性(Jaccard similarity)
  - $SIM(S,T)=\frac{|S\cap T|}{|S\cup T|}$



计算问题：给定一族集合$\mathcal{S}$（一组文档），计算$\mathcal{S}$中所有相似性≥ s的集合对。

$|S|=8, |T|=5, |S\cup T|=10, |S\cap T|=3$, SIM$(S,T)=\frac{|S\cap T|}{|S\cup T|}=\frac{3}{10}$

问题5：LSH技术 43

- **相似文档的计算问题**：给定一族集合$\mathcal{S}$（一组文档），计算$\mathcal{S}$中所有相似性≥ s的集合对。
- 例子 $U=\{\text{Cruise, Ski, Resorts, Safari, Stay@Home}\}$

$S_1=\{\text{Cruise, Safari}\}$　$S_3=\{\text{Ski, Safari, Stay@Home}\}$

$S_2=\{\text{Resorts}\}$　$S_4=\{\text{Cruise, Resorts, Safari}\}$

- 计算问题
  - 给定$\mathcal{S}=\{S_1,S_2,S_3,S_4\}$及$s=\frac{1}{2}$，输出所有相似性不小于$\frac{1}{2}$的集合对

- **Shingle集合可能非常大**
- **两两计算相似性$O(N^2)$**
- **需要更高效的数据技术：minHash+LSH**

问题5：LSH技术 44

- 例子 $U = \{\text{Cruise, Ski, Resorts, Safari, Stay@Home}\}$

$S_1 = \{\text{Cruise, Safari}\}$ $S_3 = \{\text{Ski, Safari, Stay@Home}\}$

$S_2 = \{\text{Resorts}\}$ $S_4 = \{\text{Cruise, Resorts, Safari}\}$

- 计算问题：所有相似性不小于$\frac{1}{2}$的集合对
- ①集合的矩阵表示

	S_1	S_2	S_3	S_4
Cruise	1	0	0	1
Ski	0	0	1	0
Resorts	0	1	0	1
Safari	1	0	1	1
Stay@Home	0	0	1	0

问题5：LSH技术 45

- 例子 $U = \{\text{Cruise, Ski, Resorts, Safari, Stay@Home}\}$

$S_1 = \{\text{Cruise, Safari}\}$ $S_3 = \{\text{Ski, Safari, Stay@Home}\}$

$S_2 = \{\text{Resorts}\}$ $S_4 = \{\text{Cruise, Resorts, Safari}\}$

- 计算问题：所有相似性不小于$\frac{1}{2}$的集合对
- ②minHash: 计算文档的签名；利用随机排列(Permutation)
  - 考虑一个排列π: 01234 → 40312，利用排列定义minHash

		S_1	S_2	S_3	S_4
0	Cruise	1	0	0	1
1	Ski	0	0	1	0
2	Resorts	0	1	0	1
3	Safari	1	0	1	1
4	Stay@Home	0	0	1	0

		S_1	S_2	S_3	S_4
0(1)	Ski	0	0	1	0
1(3)	Safari	1	0	1	1
2(4)	Stay@Home	0	0	1	0
3(2)	Resorts	0	1	0	1
4(0)	Cruise	1	0	0	1

- **minHash值**(签名)：S_i的签名是**最小**的值非0的**行号**
- $h(S_1) = 1$， $h(S_2) = 3$， $h(S_3) = 0$， $h(S_4) = 1$

问题5：LSH技术 46

- minHash的性质
  - 两个集合S_i和S_j对应的minHash值满足$h(S_i) = h(S_j)$的概率与两者的Jaccard相似性取值相等，即

$$\mathbb{Pr}[h(S_i) = h(S_j)] = SIM(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$$

- **注意：minHash函数h是基于一个随机选取的排列**
- **不同排列会不同的h**

1	1	→	x	4	Cruise	1	0	0	1

证明

- (1)两列都为0的行号不可能成为minHash值，其它行号等可能
- (2)$\mathbb{Pr}[h(S_1) = h(S_4)] = \frac{x}{x+y} = \frac{|S_1 \cap S_4|}{S_1 \cup S_4} = SIM(S_1, S_4)$

问题5：LSH技术 47

- 例子 $U = \{\text{Cruise, Ski, Resorts, Safari, Stay@Home}\}$

$S_1 = \{\text{Cruise, Safari}\}$ $S_3 = \{\text{Ski, Safari, Stay@Home}\}$

$S_2 = \{\text{Resorts}\}$ $S_4 = \{\text{Cruise, Resorts, Safari}\}$

- 计算问题：所有相似性不小于$\frac{1}{2}$的集合对
- ③minHash签名: 利用多个函数构造
  - $h(S_1) = 1$， $h(S_2) = 3$， $h(S_3) = 0$， $h(S_4) = 1$
  - 11个集合
  - 12个minHash函数

S_1	S_2	S_3	S_4
1	3	0	1

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
2	2	1	0	0	1	3	2	5	0	3
1	3	2	0	2	2	1	4	2	1	2
3	0	3	0	4	3	2	0	0	4	2
0	4	3	1	5	3	3	2	3	5	4
2	1	1	0	4	1	2	1	4	2	5
4	2	1	0	5	2	3	2	3	5	4
2	4	3	0	5	3	3	4	4	5	3
0	2	4	1	3	4	3	2	2	2	4
0	2	1	0	5	1	1	1	1	5	1
0	5	1	0	2	1	3	2	1	5	4
1	3	1	0	5	2	3	3	6	3	2
0	5	2	1	5	1	2	2	6	5	4

问题5：LSH技术 48

- 局部敏感哈希技术(locality sensitive hashing)

S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
2	2	1	0	0	1	3	2	5	0	3
1	3	2	0	2	2	1	4	2	1	2
3	0	3	0	4	3	2	0	0	4	2
0	4	3	1	5	3	3	2	3	5	4
2	1	1	0	4	1	2	1	4	2	5
4	2	1	0	5	2	3	2	3	5	4
2	4	3	0	5	3	3	4	4	5	3
0	2	4	1	3	4	3	2	2	2	4
0	2	1	0	5	1	1	1	1	5	1
0	5	1	0	2	1	3	2	1	5	4
1	3	1	0	5	2	3	3	6	3	2
0	5	2	1	5	1	2	2	6	5	4

- **利用minHash函数值将集合分到不同的位置**
- **相同位置的是“相似候选”**

问题5：LSH技术 49

- 局部敏感哈希技术(locality sensitive hashing)
  - 将签名矩阵分为$b = 4$条带(band)，每条带包含$r = 3$行
  - $\{S_3, S_6, S_{11}\}$被映射到同一个位置，$\{S_2, S_{10}\}$以及$\{S_8, S_9\}$也一样

Band	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
I	2	2	1	0	0	1	3	2	5	0	3
	1	3	2	0	2	2	1	4	2	1	2
	3	0	3	0	4	3	2	0	0	4	2
II	0	4	3	1	5	3	3	2	3	5	4
	2	1	1	0	4	1	2	1	4	2	5
	4	2	1	0	5	2	3	2	3	5	4
III	2	4	3	0	5	3	3	4	4	5	3
	0	2	4	1	3	4	3	2	2	2	4
	0	2	1	0	5	1	1	1	1	5	1
IV	0	5	1	0	2	1	3	2	1	5	4
	1	3	1	0	5	2	3	3	6	3	2
	0	5	2	1	5	1	2	2	6	5	4

问题5：LSH技术 50

- LSH的性质
  - 考虑任意两个集合S和T，$SIM(S,T)=s$，两者的minHash签名至少有一条带上所有行都相同的概率如下
    $$f(s)=1-(1-s^r)^b$$
  - 假设$SIM(S,T)=0.8$，$b=20, r=5$

Band	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
I	2	2	1	0	0	1	3	2	5	0	3
	1	3	2	0	2	2	1	4	2	1	2
	3	0	3	0	4	3	2	0	0	4	2
II	0	4	3	1	5	3	3	2	3	5	4
	2	1	1	0	4	1	2	1	4	2	5
	4	2	1	0	5	2	3	2	3	5	4
III	2	4	3	0	5	3	3	4	4	5	3
	0	2	4	1	3	4	3	2	2	2	4
	0	2	1	0	5	1	1	1	1	5	1
IV	0	5	1	0	2	1	3	2	1	5	4
	1	3	1	0	5	2	3	3	6	3	2
	0	5	2	1	5	1	2	2	6	5	4

035

1/000

- 假

选

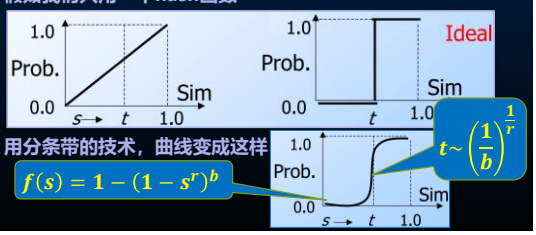
问题5：LSH技术 51

- LSH的性质
  - 考虑任意两个集合S和T，$SIM(S,T)=s$，两者的minHash签名至少有一条带上所有行都相同的概率如下
    $$f(s)=1-(1-s^r)^b$$

(b,r) $f(s)=1-(1-s^r)^b$ ↘	$(4,3)$	$(16,4)$	$(20,5)$	$(25,5)$	$(100,10)$
$s=0.2$	0.0316	0.0252	0.0063	0.0079	0.0000
$s=0.4$	0.2324	0.3396	0.1860	0.2268	0.0104
$s=0.5$	0.4138	0.6439	0.4700	0.5478	0.0930
$s=0.6$	0.6221	0.8914	0.8019	0.8678	0.4547
$s=0.8$	0.9432	0.9997	0.9996	0.9999	0.9999
$s=1.0$	1.0	1.0	1.0	1.0	1.0
Threshold $t=(\frac{1}{b})^{(\frac{1}{r})}$	0.6299	0.5	0.5492	0.5253	0.6309

问题5：LSH技术 52

- LSH的性质
  - 考虑任意两个集合S和T，$SIM(S,T)=s$，两者的minHash签名至少有一条带上所有行都相同的概率如下
    $$f(s)=1-(1-s^r)^b$$
- **假如我们只用一个hash函数**

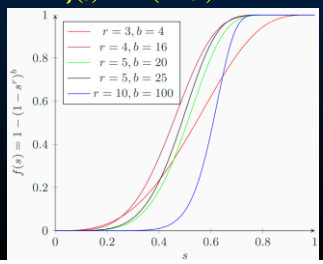


- **用分条带的技术，曲线变成这样**

$$f(s)=1-(1-s^r)^b$$

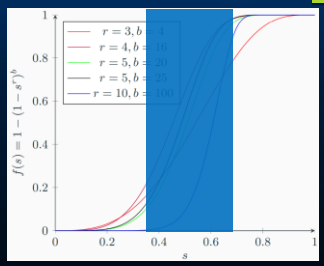
问题5：LSH技术 53

- LSH的性质
  - 考虑任意两个集合S和T，$SIM(S,T)=s$，两者的minHash签名至少有一条带上所有行都相同的概率如下
    $$f(s)=1-(1-s^r)^b$$



问题5：LSH技术 54

- LSH的性质



- $SIM(S_1,S_2)\geq r_1 \Rightarrow \mathbb{Pr}[h(S_1)=h(S_2)]\geq p_1$
- $SIM(S_1,S_2)\leq r_2 \Rightarrow \mathbb{Pr}[h(S_1)=h(S_2)]\leq p_2$

问题5：LSH技术 55

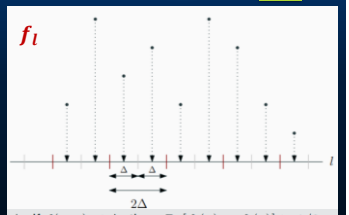
相似性文档的算法(LSH-Based)

- ①将文档转换为集合；
- ②利用minHash计算所有文档的签名
- ③根据签名，将文档映射到不同的位置(桶)
  - 一个文档会被映射b次
- ④扫描每一个桶，仅考虑桶内文档相互之间的相似性，计算精准的Jaccard相似性，并与阈值比较，输出相似文档对
- 算法代价：$O(N^2)\rightarrow O(\alpha\cdot P)$
  - P是真实的相似文档对的数目
  - α是与b,r值相关的系数，表示候选文档对与真实相似文档对的比例关系

问题5：LSH技术

56

- LSH的应用范围
  - 海明距离
  - Cosine距离
  - 欧几里得距离 ➡
  - χ^2距离
  - ...



1. If $d(x,y) \le \Delta$, then $Pr[f_l(x) = f_l(y)] \ge 1/2$.
2. If $d(x,y) > 4\Delta$, then $Pr[f_l(x) = f_l(y)] \le 1/3$.

60

问题2：不重复元素数

61

流模型（Streaming Model）

- 给定数据流 $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$，其中$a_i \in [m]$
- 对应的频次向量为

$$f = (f_1, \ldots, f_m), s.t. \sum_{1 \le i \le m} f_i = n.$$

定义 [The Distinct Elements Problem]

给定数据流σ，计算$\sum_{1 \le i \le m} \mathbf{I}[f_i > 0]$。

$$\mathbf{I}[f_i > 0] = 1 \Leftrightarrow f_i > 0$$

问题2：不重复元素数

62

精确计算需要多少空间代价？

- 方法1： $O(m)$ 位.
  - 为[m]中每个元素维护1个bit, 长度为m的向量
- 方法2： $O(n \log m)$ 位.
  - 维护n个数，每一个使用$\log\ m$位.

问题2：不重复元素数

63

- 一个理想化的方案：假设可以存储实数，真实值为D
- 利用哈希函数（hash function）
  - h : [m] ⟼ [0, 1]
  - h(i)的函数值是[0, 1]实数，均匀分布

FM算法[Flajolet-Martin 1985]

```
/** 维护变量z **/
1. 令z = 1，随机选取哈希函数h : [m] ⟼ [0, 1];
2. for 每一个元素i do
3.     z = min{z, h(i)};
4. return 1/z − 1.
```

问题2：不重复元素数

64

- FM算法的分析：令算法结果$X = \frac{1}{z} - 1$
- $\mathbb{E}[z] = \frac{1}{D+1}$， $\mathbf{var}[z] \le \frac{2}{(D+1)(D+2)}$

$$\Pr\left[\left|z - \frac{1}{D+1}\right| > c\frac{1}{D+1}\right] < \frac{2(D+1)^2}{c^2 \cdot (D+1)(D+2)} < \frac{2}{c^2}$$

$$\Pr[|X - D| > \epsilon D] \le \Pr\left[\left|z - \frac{1}{D+1}\right| > \frac{\epsilon D}{(D+1+\epsilon D)}\frac{1}{(D+1)}\right]$$

令$c = \frac{\epsilon D}{D+1+\epsilon D}$

$$< \frac{2(D+1+\epsilon D)^2}{\epsilon^2 D^2} = 2\left(\frac{D+1}{\epsilon D} + 1\right)^2 < 2\left(\frac{2}{\epsilon} + 1\right)^2$$

问题2：不重复元素数 65

利用多次运行

FM+ Algorithm

/** Maintain a variable z **/

1. **for** j from 1 to k
2. 随机选取一个哈希函数h_j : [m] ⟼ [0, 1]
3. z_j = 1.
4. 每次遇到 i，更新: $z_j = \min(z_j, h_j(i))$
5. $Z = \frac{1}{k}\sum_{j=1}^{k} z_j$;
6. **return** 1/Z − 1.

问题2：不重复元素数 66

- FM+算法分析：$X = \frac{1}{Z} - 1$，$Z = \frac{1}{k}\sum_{j=1}^{k} z_j$
- $\mathbb{E}[Z] = \mathbb{E}[z]$，$\mathbf{var}[Z] = \frac{\mathbf{var}[Z]}{k}$

$$\Pr\left[\left|Z - \frac{1}{D+1}\right| > c\frac{1}{D+1}\right] < \frac{(D+1)^2}{c^2}\mathbf{var}[Z]$$

令$c = \frac{\epsilon D}{D+1+\epsilon D} \Rightarrow$

$$\Pr[|X - D| > \epsilon D] < \frac{2(D+1+\epsilon D)^2}{k\epsilon^2 D^2} < \frac{2}{k}\left(\frac{2}{\epsilon}+1\right)^2$$

- 这里，ϵ为精度要求，假设概率要求为$1-\delta$，只需

$$\frac{2}{k}(\frac{2}{\epsilon}+1)^2 < \delta$$

$$\Rightarrow k > \frac{2}{\delta}\left(\frac{2}{\epsilon}+1\right)^2 = O(\frac{1}{\epsilon^2\delta})$$

问题2：不重复元素数 67

利用Median技术

FM++算法

1. 运行具有常数概率的FM+算法$t = \Theta(\log\frac{1}{\delta})$次;
2. $\hat{D}$为所有 t 个结果的中间值;
3. **return** $\hat{D}$;

- FM++是以1 − δ概率保证(1 ± ϵ)近似的算法
- 整体代价为$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$个实数的存储空间

问题2：不重复元素数 68

然而，我们无法存储实数，需要更实际的算法

- 设计一个$O(\log m)$位空间代价的$O(1)$近似算法

$$\frac{1}{C} \times D \le \hat{D} \le C \times D$$

Practical FM算法

1. 随机选取**2-相对独立哈希函数**h: $[m] \mapsto [m]$;
2. $z = 0$;
3. **for** 每一个元素j **do**

$$z = \max\{z, \mathbf{zeros}(h(j))\};$$

4. **return** $\hat{D} = 2^{z+1/2}$;

$$\mathbf{zeros}(p) = \max\{i | p\%2^i = 0\}$$

0	1	0	1	0	0

$\mathbf{zeros}(\cdot) = 2$

问题2：不重复元素数 69

定义[2-相对独立哈希函数]

一个从[X]到[Y]的哈希函数族$\mathcal{H}$是**2-相对独立的**，若对$\forall h \in \mathcal{H}$, $a, b \in [Y]$及$i, j \in$ [X] (满足$i \neq j$), 有

$$\Pr(h(i) = a \wedge h(j) = b) = \frac{1}{Y^2}$$

定义[k-相对独立哈希函数]

哈希函数族$\mathcal{H}$是***k*-相对独立的**，若对$\forall h \in \mathcal{H}$, $j_1, \dots, j_k \in [Y]$及$i_1, \dots, i_k \in [X]$ (满足$i_x \neq i_y$), 有

$$\Pr(h(i_1) = j_1 \wedge \cdots \wedge h(i_k) = j_k) = 1/Y^k$$

- 理论上，可以利用$\log_2|\mathcal{H}|$位存储某个$h \in \mathcal{H}$

问题2：不重复元素数 70

2-相对独立哈希函数族的例子

假设Y是一个素数(若不是，可以取比Y大的第一个素数), 随机选$p, q \in$ {0,1,2,…, Y-1}, 函数族如下

$$\mathcal{H} = \{h_{p,q}(x) = (px + q)\%Y\}$$

- 显然，存储一个h只需$O(\log Y)$位的空间代价

k-相对独立哈希函数的例子

- $\mathcal{H}_{poly}$：至多为k-1阶多项式, 系数在$[n]$中, n为素数, $|\mathcal{H}_{poly}| = n^k$, 存储需$O(k \log n)$位
- 存储2-相对独立$\mathcal{H}_{poly}$某个函数,需$O(2 \log n)$位

问题2：不重复元素数

71

Practical FM算法

1. 随机选取**2-相对独立哈希函数**h: $[m] \mapsto [m]$;
2. $z = 0$;
3. **for** 每一个元素j **do**
   $z = \max\{z, \text{zeros}(h(j))\}$;
4. **return** $\hat{D} = 2^{z+1/2}$;

存储哈希函数需$O(\log m)$位；存储z需$\log\log m$位

定理 Practical FM算法以$1 - \frac{2\sqrt{2}}{c}$概率满足$\frac{D}{c} \le \hat{D} \le C \times D$。

问题2：不重复元素数

72

定理 Practical FM算法以$1 - \frac{2\sqrt{2}}{c}$概率满足$\frac{D}{c} \le \hat{D} \le C \times D$。

- 令$X_{r,j} = 1 \Leftrightarrow$ zeros(h(j)) $\ge r$，其中$j \in [m]$且$r \ge 0$
- 令$Y_r = \sum_{j \in \sigma} X_{r,j} = \sum_{j:f_j>0} X_{r,j}$, Y_0即是不重复元素数目
  - Y_r：哈希值二进制尾部**0的数目** $\ge r$的元素数
- 假设算法结束时，$z = t$
  - $t \ge r \Leftrightarrow \exists j$有zeros(h(j)) $\ge r \Leftrightarrow X_{r,j} = 1 \Leftrightarrow Y_r > 0$
  - $t < r \Leftrightarrow \forall j$有zeros(h(j)) $< r \Leftrightarrow X_{r,j} = 0 \Leftrightarrow Y_r = 0$

Median技术⇒1−δ概率保证$O(1)$近似，空间$O(\log\frac{1}{\delta}\log m)$

Practical FM: ❶$z = \max\{z, \text{zeros}(h(j))\}$; ❷ $\hat{D} = 2^{z+1/2}$;

问题2：不重复元素数

73

- 两层hash，$O(\log m + \frac{1}{\epsilon^2}(\log\frac{1}{\epsilon} + \log\log m))$空间算法

BJKST 算法

1. 随机选2-相对独立哈希函数h : [m] → [m];
2. 随机选2-相对独立哈希函数g : [m] → $[b\epsilon^{-4}\log^2 m]$;
3. $z = 0$, B = ∅;
4. **if** zeros(h(j)) ≥ z **then**
5. $B = B \cup \{(g(j), zeros(h(j)))\}$;
6. **if** $|B| > \frac{c}{\epsilon^2}$ **then**
7. $z = z + 1$，从B中删除(α, β)，其中β < z;
8. **return** $\hat{D} = |B|2^z$;

问题2：不重复元素数

74

定理[BJKST算法性能]

BJKST算法使用$O(\log m + \frac{1}{\epsilon^2}(\log\frac{1}{\epsilon} + \log\log m))$空间，可以实现至少2/3概率保证$(1 \pm \epsilon)$近似。

- 存储h需$O(\log m)$
- 存储g需$O(\log(b\epsilon^{-4}\log^2 m))$
- 存储|B|个g取值需$O(\epsilon^{-2}\log(b\epsilon^{-4}\log^2 m))$
- 存储|B|个zeros取值需$O(\epsilon^{-2}\log\log m)$
- 因此，空间代价为$O(\log m + \frac{1}{\epsilon^2}(\log\frac{1}{\epsilon} + \log\log m))$
- Ziv Bar-Yossef et al. Counting Distinct Elements in a Data Stream. In: *RANDOM 2002*

75

👉 亚线性时间大数据算法

问题5：点查询

76

流模型（The Streaming Model）

- 给定数据流$\sigma = < a_1, a_2, \ldots, a_n >$，其中$a_i \in [m]$
- 对应的频次向量为

$$f = (f_1, \ldots, f_m), s.t. \sum_{1 \le i \le m} f_i = n.$$

定义[点查询(point query)问题]

给定数据流 σ 和事先未知查询 a_i，输出 f_i。

问题5：点查询 77

ℓ_p范数：$\|x\|_p = (\sum_i |x_i|^p)^{1/p}$

定义1.10[ℓ_p -近似点查询（频度估计）]

给定数据流 σ 以及事先未知查询 a_i 输出 $\hat{f}_i$ 满足 $\hat{f}_i \in f_i \pm \epsilon\|f\|_p$。

- $\|x\|_1 \geq \|x\|_2 \geq \cdots \geq \|x\|_\infty$，p越大，估计越准确
- $\|x\|_0 = m$，是不同元素的数目
- $\|x\|_1 = n$，是数据流的长度
- $\|x\|_\infty = \max\{f_i\}$，是最大频度

问题5：点查询–ℓ_1近似 78

Misra-Gries 算法[1982]

/** Maintain a set A consists of pairs $(i, \hat{f}_i)$ **/

1. A ← ∅;
2. 对每一个数据流中的元素e,
   (a) 如果 e ∈ A, 令 $(e, \hat{f}_e) \leftarrow (e, \hat{f}_e + 1)$
   (b) 否则，如果 |A| < 1/ϵ , 将 (e, 1) 插入 A
   (c) 否则，将所有 A 中计数减1
   $(j, \hat{f}_j) \leftarrow (j, \hat{f}_j - 1)$,
   如果 $\hat{f}_j$ = 0, 从 A 中删除 (j, 0)
3. 对于查询 i, 如果 i ∈ A, **return** $\hat{f}_i$, 否则 **return** 0;

问题5：点查询–ℓ_1近似 79

定理1.17

Misra-Gries算法空间代价为$O(\epsilon^{-1}\log mn)$，对任意查询 i，返回 $\hat{f}_i$ 满足

$$f_i - \epsilon n \leq \hat{f}_i \leq f_i$$

- 如果不发生减1的情况，那么$\hat{f}_i = f_i$
- 如果发生了减1的情况，有$\hat{f}_i < f_i$
- 假设共发生了c次减1的情况，总数减少$\frac{c}{\epsilon} \leq n$
- 每个计数至多减少c， $\hat{f}_i \geq f_i - c \geq f_i - \epsilon n$

问题5：点查询–ℓ_1近似 80

Metwally 算法[2005]

/** Maintain a set A consists of pairs $(i, \hat{f}_i)$ **/

1. A ← ∅;
2. 对每一个数据流中的元素e,
   (a) 如果 e ∈ A, 令 $(e, \hat{f}_e) \leftarrow (e, \hat{f}_e + 1)$
   (b) 否则，如果|A| < 1/ϵ，将 (e, 1) 插入 A
   否则，将 (e, MIN + 1) 插入 A，并删除一个满足$\hat{f}_{e'}$ = MIN = min$\{\hat{f}_i\}$的元素
3. 查询 i, 如果 i ∈ A, **return** $\hat{f}_i$, 否则 **return** MIN;

问题5：点查询–ℓ_1近似 81

定理1.18

Metwally算法空间代价为 $O(\epsilon^{-1}\log mn)$，对任意查询 i，返回 $\hat{f}_i$ 满足

$$f_i \leq \hat{f}_i \leq f_i + \epsilon n$$

- 如果不发生删除的情况，那么$\hat{f}_i = f_i$
- 如果删除，计数一定不大于删除后的MIN，有$\hat{f}_i \geq f_i$
- A中元素和总是n，$MIN\frac{1}{\epsilon} \leq n \Rightarrow MIN \leq \epsilon n$
- 每个元素至多超出真实值MIN， $\hat{f}_i \leq f_i + \epsilon n$

问题5：点查询–ℓ_1近似 82

支持删除的流模型（turnstile）

- 给定数据流$\sigma = < \pm a_1, \pm a_2, \ldots, \pm a_n >$，其中$a_i \in [m]$
- 对应的频次向量为
  $$f = (f_1, \ldots, f_m)$$
- 初始化向量$f \in \mathbb{R}^m$为0
- 每次更新 (i, c)，使得 $f_i \leftarrow f_i + c$， f_i可以比0小
- c = 1 即是之前所用的普通流模型（Vanilla）
- c > 0 被称作Cash Register流模型

问题5：点查询–ℓ_1近似 83

Turnstile模型

Count 算法[2005]

1. k = $\frac{b}{\epsilon}$, C[1 . . . k] ← 0;
2. 随机选择一个2-相对独立哈希函数h : [m] → [k];
3. **for** 每一次更新 (j, c) **do**
4. C[h(j)] = C[h(j)] + c;
5. **return** $\hat{f}_a$ = C[h(a)] 对查询元素 a;

▶ 空间代价O($\frac{b}{\epsilon} \cdot \log n$)

▶ **Pr**[$|\hat{f}_a - f_a| \geq \epsilon \|f_{-a}\|_1$] ≤ $\frac{1}{b}$, 其中$\|f_{-a}\|_1 = \|f\|_1 - |f_a|$

问题5：点查询–ℓ_1近似 84

Cash Register模型：c > 0

Count-Min 算法[2005]

1. C[1 . . . t][1 . . . k] ← 0, k = $\frac{2}{\epsilon}$ 且 t = $\lceil \log \frac{1}{\delta} \rceil$;
2. 随机选择 t 个2-相对独立哈希函数h_i : [m] → [k];
3. **for** 每一次更新 (j, c) **do**
4. **for** i = 1 to t **do**
5. C[i][h_i(j)] = C[i][h_i(j)] + c;
6. **return** $\hat{f}_a = min_{1 \leq i \leq t} C[i][h_i(a)]$ 对查询元素 a;

▶ 空间代价O($\frac{1}{\epsilon} \log \frac{1}{\delta} \cdot \log n$)

问题5：点查询–ℓ_1近似 85

定理1.19[Count-Min]

Count-Min 算法以 1 − δ 概率给出ℓ_1近似–点查询的(1 + ϵ)近似，具体地

$$f_a \leq \hat{f}_a \leq f_a + \epsilon \|f_{-a}\|_1$$

这里，$\|f_{-a}\|_1 = \|f\|_1 - |f_a|$。

问题5：点查询–ℓ_1近似 86

Turnstile模型

Count-Median 算法[2005]

1. C[1 . . . t][1 . . . k] ← 0, k = $\frac{2}{\epsilon}$ 且 t = $\lceil \log \frac{1}{\delta} \rceil$;
2. 随机选择 t 个2-相对独立哈希函数h_i : [m] → [k];
3. **for** 每一次更新 (j, c) **do**
4. **for** i = 1 to t **do**
5. C[i][h_i(j)] = C[i][h_i(j)] + c;
6. **return** $\hat{f}_a = median_{1 \leq i \leq t} C[i][h_i(a)]$ 对查询 a;

▶ 空间代价O($\frac{1}{\epsilon} \log \frac{1}{\delta} \cdot \log n$)

问题5：点查询–ℓ_1近似 87

定理1.19[Count-Median]

Count-Median 算法以 1 − δ 概率给出ℓ_1近似–点查询的(1 + ϵ)近似，具体地

$$|\hat{f}_a - f_a| \leq \epsilon \|f_{-a}\|_1$$

问题5：点查询–ℓ_2近似 88

Turnstile模型，质量更好的估计结果

CountSketch 算法[2004]

1. C[1 . . . t]← 0, k = $\frac{3}{\epsilon^2}$;
2. 随机选择 1 个2-相对独立哈希函数 h : [m] → [k];
3. 随机选择 1 个2-相对独立哈希函数 g : [m] → {-1,1};
4. **for** 每一次更新 (j, c) **do**
5. C[h(j)] = C[h(j)] + c · g(j);
6. **return** $\hat{f}_a$ = g(a) · C[h(a)] 对查询 a;

▶ 空间代价O($\frac{1}{\epsilon^2} \cdot \log n$)

▶ 针对Turnstile模型，以常数2/3概率，有(1 ± ϵ)近似比

问题5：点查询–ℓ_2近似

89

利用Median技术改进估计质量

CountSketch+ 算法

1. C[1 . . . t][1 . . . k] ← 0, k = $\frac{3}{\epsilon^2}$ 且 t = O(log $\frac{1}{\delta}$);
2. 随机选择 t 个2-相对独立哈希函数h_i : [m] → [k];
3. 随机选择 t 个2-相对独立哈希函数g_i : [m] → {-1,1};
4. **for** 每一次更新 (j, c) **do**
5. **for** i = 1 to t **do**
6. C[i][h_i(j)] = C[i][h_i(j)] + c · g_i(j);
7. **return** $\hat{f_a}$ = $median_{1\le i\le t} g_i(a)C[i][h_i(a)]$ 对查询 a;

- 空间代价O($\frac{1}{\epsilon^2}\log\frac{1}{\delta}\cdot\log n$)

问题5：点查询–ℓ_2近似

90

$t = \log\frac{1}{\delta}$使得切尔诺夫不等式

定理1.21[CountSketch+]

CountSketch+ 算法至少以 1 − δ 概率给出ℓ_2近似–点查询的(1 + ϵ)近似估计，具体地

$$|\hat{f_a} - f_a| \le \epsilon\|f_{-a}\|_2$$

问题5：点查询

91

点查询（频度估计）问题的算法

Method	$\hat{f_a} - f_a \in$	Space	Pr	Mod.
Misra-Gries	$[-\epsilon\|f_{-a}\|_1, 0]$	$O\left(\frac{\log mn}{\epsilon}\right) = O(\frac{\log n}{\epsilon})$	0	+
Metwally	$[0, \epsilon\|f_{-a}\|_1]$	$O\left(\frac{\log mn}{\epsilon}\right) = O(\frac{\log n}{\epsilon})$	0	+
CountSketch+	$\pm\epsilon\|f_{-a}\|_2$	$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\cdot\log n)$	δ	±
Count-Min	$[0, \epsilon\|f_{-a}\|_1]$	$O(\frac{1}{\epsilon}\log\frac{1}{\delta}\cdot\log n)$	δ	+
Count-Median	$\pm\epsilon\|f_{-a}\|_1$	$O(\frac{1}{\epsilon}\log\frac{1}{\delta}\cdot\log n)$	δ	±

- m < n，+表示Cash Register Model，±表示Turnstile Model

问题6：频度矩估计

92

假设我们现在处理Vanilla Model

定义1.13[频度矩--Frequency Moments]

给定数据流 $\sigma =< a_1, \dots, a_n >$，假设$a_i \in$ [m]，令f = f(σ) = $(f_1, f_2, \dots, f_m)$，显然，$\sum f_i = n$。数据流 σ 的第 k 阶频度矩（kth frequency moment）表示为 $F_k(\sigma)$ 或者 F_k，

$$F_k = \sum_{i=1}^{m} |f_j|^k = \|f\|_k^k$$

- F_0是不重复元素的数目
- F_1是计数问题
- F_2可表示关系数据自连接（self join）操作结果大小
- 这部分为F_k(k ≥ 2)估计问题提供亚线性空间的算法

问题6：频度矩估计

93

- AMS采样处理的一般问题：给定函数g满足g(0) = 0，估计$\sum_{i\in[m]} g(f_i)$，这里$g(f_i)$是定义在频度f_i上的某个函数。
- 采样算法：在区域[n]上均匀采样得到j，获取某个元素 a_j ，然后计算$r = |\{i \ge j | a_i = a_j\}|$；最后，输出$X = n(g(r) - g(r-1))$。

$$\mathbb{E}[X] = \sum_b \mathbf{Pr}[a_j = b]\,\mathbb{E}[X|a_j = b]$$

$$= \sum_b \frac{f_b}{n}\left(\sum_{r=1}^{f_b}\frac{n(g(r) - g(r-1))}{f_b}\right) = \sum_b g(f_b) = \sum_{b\in[m]} g(f_b)$$

- 因为$F_k = \sum_{i\in[m]}(f_i)^k$，AMS采样器为$X = n\left(r^k - (r-1)^k\right)$，$\mathbb{E}[X] = F_k$

问题6：频度矩估计

94

Basic AMS 算法

1. (L,r, a) ← (0, 0, 0);
2. **for** 第 j 个元素 a_j **do**
3. L ← L + 1;
4. β ← random bit with **Pr**[β = 1] = $\frac{1}{L}$;
5. **if** β = 1 **then**
6. a ← a_j , r ← 0;
7. **if** a_j== a **then**
8. r ← r + 1;
9. **return** $X = L(r^k - (r-1)^k)$;

问题6：频度矩估计

95

空间代价

- 存储 L 和 r 需要 log n 位
- 存储 a 需要 log m 位
- 共计 O(log m + log n) 位代价

直观思想

- 随机选取一个位置 y ∈ [n]
- $a = a_y$
- $r = |\{j | j \geq y 并且 a_j = a_y\}|$

$$\mathbf{Pr}\left[|X - \mathbb{E}[X]| \geq \epsilon \mathbb{E}[X]\right] \leq \frac{km^{1-\frac{1}{k}}F_k^2}{\epsilon^2 F_k^2} = \frac{km^{1-\frac{1}{k}}}{\epsilon^2}$$

问题6：频度矩估计

96

Final AMS 算法

1. 利用Median-of-Mean技术调用Basic AMS算法;
2. 计算$t = c\log\frac{1}{\delta}$个平均值;
3. 每个平均值是$r = \frac{3k}{\epsilon^2}\cdot m^{1-\frac{1}{k}}$次调用的平均值;
4. **return** t个数值的中间值;

- $\Pr\left[|X - \mathbb{E}[X]| \geq \epsilon\mathbb{E}[X]\right] \leq \frac{km^{1-\frac{1}{k}}}{r\epsilon^2} = \frac{1}{3} \Rightarrow \delta$
- 空间代价：

$$tr \cdot O(\log m + \log n) = O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\cdot km^{1-\frac{1}{k}}(\log m + \log n))$$

问题6：频度矩估计

97

该算法在k = 2时，代价为$\tilde{O}(\epsilon^{-2}n^{0.5})$，过大

Basic F_2 AMS 算法(Tug-of-War Sketch)

1. 随机选择4-相对独立哈希函数h : [m] → {−1, 1};
2. x ← 0;
3. **for** 每一次更新 (j, c) **do**
4. x ← x + c · h(j);
5. **return** x^2;

- $\mathbb{E}[X^2] = F_2$； $\mathbf{Var}[X^2] \leq 2F_2^2$
- 令$t = c\log\frac{1}{\delta}$且$r = \frac{3\mathbf{Var}[X^2]}{\epsilon^2\mathbb{E}[X^2]^2} \leq \frac{6}{\epsilon^2} \Rightarrow$ (ϵ, δ)-近似算法
- 空间代价$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log n)$

问题7：频繁元素

98

定义1.14[频繁元素，Frequent Items, heavy hitters]

给定数据流 σ 和参数 k，输出频繁元素集合 {j | f_j > n/k}.

利用点查询的算法求解

- 令 ϵ = 1/k，执行Misra-Gries 算法，输出A中元素

Misra-Gries 算法[1982]

1. A ← ∅;
2. 对每一个数据流中的元素e,
   (a) 如果 e ∈ A, 令 $(e, f_e) \leftarrow (e, f_e + 1)$
   (b) 否则，如果 |A| < 1/ϵ , 将 (e, 1) 插入 A
   (c) 否则，将所有 A 中计数减1
   $(j, \hat{f}_j) \leftarrow (j, \hat{f}_j - 1)$,
   如果 $\hat{f}_j = 0$, 从 A 中删除 (j, 0)
3. 对于查询 i, 如果 i ∈ A, **return** $\hat{f}_i$, 否则 **return** 0;

- $f_j > \frac{n}{k} \Rightarrow \hat{f}_j > 0 \Rightarrow j \in A$

问题7：频繁元素

99

- 想估计最大频度，是否可以在亚线性代价内实现?
  - 考虑一个hard情况，每个元素频度都差不多，最后一个元素决定最大频度是多少
- 放松要求：最频繁的元素是heavy的，设计算法

定义1.15[(α, p)-Heavy Hitters]

给定参数 α ∈ (0, 1) 和 p ∈ [1, ∞)，对于数据流 σ，假设对应的频度向量为 f，(α, p)-Heavy Hitters定义如下

$$HH_\alpha^p(f) = \{i \in [m]: f_i \geq \alpha\|f\|_p\}$$

定义1.16[α-Heavy Hitters]

元素 i 是 α-heavy的，如果$f_i \geq \alpha\|f\|_1 = \alpha\sum f_j$。上述元素 i 构成的集合表示为$HH_\alpha$。

问题7：频繁元素

100

定义1.17[Approximate α-Heavy Hitters]

给定参数 α, ϵ ∈ (0, 1)，对于数据流 σ，假设对应的频度向量为 f，计算一个集合 S ⊆ [m] 使得

$$HH_\alpha \subseteq S \subseteq HH_{(1-\epsilon)\alpha}$$

利用ℓ_1点查询算法求解，设计(ϵ, δ)近似算法

- 令 $\epsilon^2 = \epsilon\alpha/2$ 且 $\delta' = \delta/m$，用Count-Median实现(ϵ', δ')近似点查询算法，为每个元素 i ∈ [m] 估计$\hat{f}_i$
- 返回集合 $S = \{i \in [m] : \hat{f}_i \geq (\alpha - \epsilon\alpha/2)\|f\|_1\}$

- 算法空间代价：$O(\frac{1}{\alpha}\frac{1}{\epsilon}\left(\log\frac{1}{\delta} + \log m\right)\log n)$

问题7：频繁元素

101

① 给定元素$i \in HH_\alpha$，那么$f_i \geq \alpha\|f\|_1$

$$\hat{f}_i \geq f_i - \epsilon\alpha/2\|f\|_1 \geq (\alpha - \epsilon\alpha/2)\|f\|_1 \Rightarrow i \in S$$

② 给定元素$i \in S$，那么$\hat{f}_i \geq (\alpha - \epsilon\alpha/2)\|f\|_1$

$$\hat{f}_i \leq f_i + \epsilon\alpha/2\|f\|_1$$

$$\Rightarrow f_i \geq \hat{f}_i - \epsilon\alpha/2\|f\|_1 \geq (1-\epsilon)\alpha\|f\|_1 \Rightarrow i \in HH_{(1-\epsilon)\alpha}$$

- 至少以 $1-\delta$ 的概率，$\forall i \in [m]$ 上述①②都满足
  - $\forall i \in [m]$ ① 成立，那么有 $HH_\alpha \subseteq S$
  - $\forall i \in [m]$ ② 成立，那么有 $S \subseteq HH_{(1-\epsilon)\alpha}$