



華東師範大學

EAST CHINA NORMAL UNIVERSITY

数据科学与工程算法基础

Algorithm Foundations of Data Science and Engineering

第五章 Sketch算法

$$(1+x)^n = 1 + \frac{nx}{1!} + \frac{n(n-1)x^2}{2!} + \dots$$

课程提纲

Content

1 算法引入

2 Misra–Gries算法

3 Count Sketch

4 Count–min Sketch

课程提纲

Content

1 算法引入

2 Misra–Gries算法

3 Count Sketch

4 Count–min Sketch

课程引入

- 面向静态数据的分析与挖掘
 - 数据量大小固定
 - 存储在磁盘上的数据可多次访问
- 越来越多的动态数据场景
 - 科学大数据：天眼、大型强子对撞机等
 - 物联网设备感知数据：智慧城市应用中的视频流、传感器数据
 - 网络流量：电信交换机转发的流量包
 - 信令数据：基站接受到的移动手机信号
 -
- 快速产生海量的数据，而且持续不断地到达

数据流

- 流数据特征
 - 数据总量不受限制
 - 数据到达速度快
 - 数据到达次序不受约束
 - 除非刻意保存，否则每个数据项只能“看”一次
- 数据流可以看作一个无限的元组序列 $\sigma = \langle a_1, \dots, a_m, \dots \rangle$
- 在某个固定的时间点，一个数据流可以被视作一个长度为 d 的固定数组 $A[0, \dots, d-1]$
- 随着后续数据的不断到达，数组 A 的长度或其中的元素都可能发生变化

数据流算法

- **目标：**在有限资源约束的情形下，处理海量的实时数据流
- **限制：**无法决定随机访问数据，只能根据数据流的顺序对数据进行 p 次扫描（在很多场景下 $p = 1$ ）
- **要求**
 - **实时性：**实时、连续地输出查询结果
 - **低空间复杂度：**数据流规模理论上无限的，为了保证算法高效稳定运行，需降低算法的空间开销
 - **结果准确性：**数据规模大、速率快，因此对于一些复杂问题，不太可能通过数据的一次遍历就获得准确答案。实际应用中，往往不要求精准的查询结果
 - **适应性：**在很多应用中，涉及多个流数据的处理，需设计具备适应性的算法

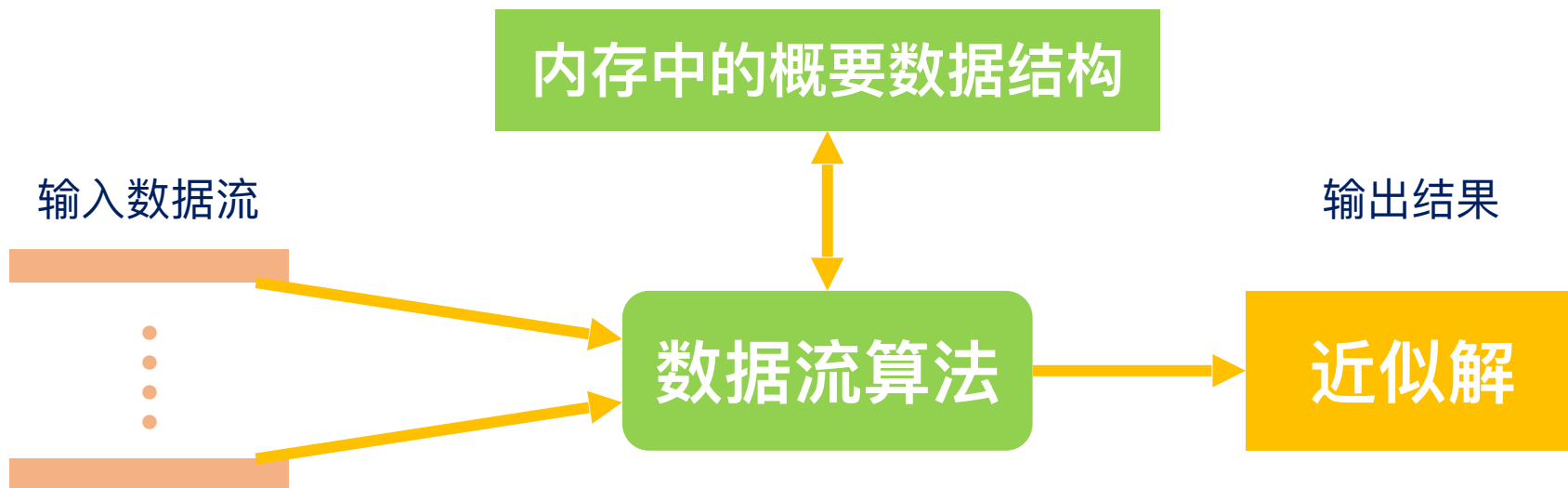
数据流算法 vs 传统算法

	传统算法	数据流算法
数据类型	有限 & 静态	无限 & 动态 & 高速
存储	硬盘	内存 & 空间限制
效率	非实时	实时 & Ad-hoc
返回值类型	精确值 / 近似值	近似值

- 数据流算法对时间、空间消耗要求高
 - 空间消耗是次线性的，或者与流的大小无关
 - 时间消耗是次线性的，或者与流的大小无关

概要数据结构

- 数据流 σ 的概要数据结构记为 $C(\sigma)$
 - 表示对数据流 σ 的压缩表示
 - 与求解问题有关



近似与随机算法

- 由于存储空间和时间的限制，在数据流上做到精确计算非常困难
- 近似算法：寻找误差在一定范围内的近似解
 - 例如，寻找一个误差在 10% 以内的解
 - 近似解落在真实值的 $(1 \pm \epsilon)$ 范围内，其中 $\epsilon = 0.1$
- 随机算法：比近似算法更宽松，允许小概率失败（解不在误差范围内）
 - 例如，估计值有 $1/100$ 的概率不在误差范围内
 - 成功的概率为 $1 - \delta$ ，其中 $\delta = 0.01$
- 随机算法： (ϵ, δ) -近似算法

近似算法

- 给定输入的流数据 σ 和准确值 $\phi(\sigma)$ ，近似算法的输出结果记为 $\mathcal{A}(\sigma)$
- ϵ -近似算法（相对误差）：如果该算法的输出结果满足
$$|\mathcal{A}(\sigma) - \phi(\sigma)| < \epsilon \phi(\sigma)$$
- ϵ -近似算法（绝对误差）：如果该算法的输出结果满足
$$|\mathcal{A}(\sigma) - \phi(\sigma)| < \epsilon$$

随机算法

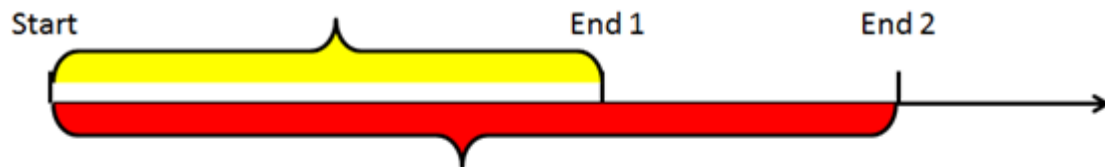
- 给定输入的流数据 σ 和准确值 $\phi(\sigma)$, 近似算法的输出结果记为 $\mathcal{A}(\sigma)$
- (ϵ, δ) -近似算法（相对误差）：如果该算法的输出结果满足

$$\Pr[|\mathcal{A}(\sigma) - \phi(\sigma)| < \epsilon\phi(\sigma)] > 1 - \delta$$

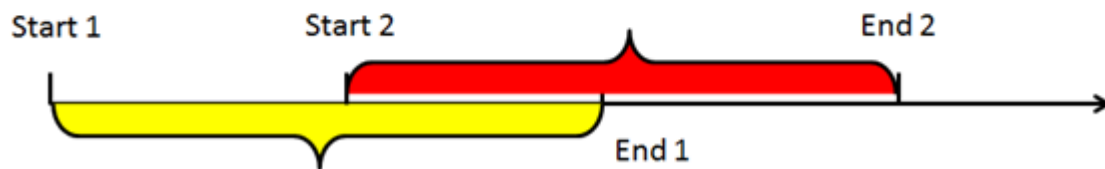
- (ϵ, δ) -近似算法（绝对误差）：如果该算法的输出结果满足

$$\Pr[|\mathcal{A}(\sigma) - \phi(\sigma)| < \epsilon] > 1 - \delta$$

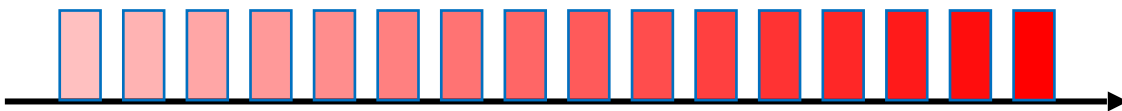
数据流模型



- 界标模型

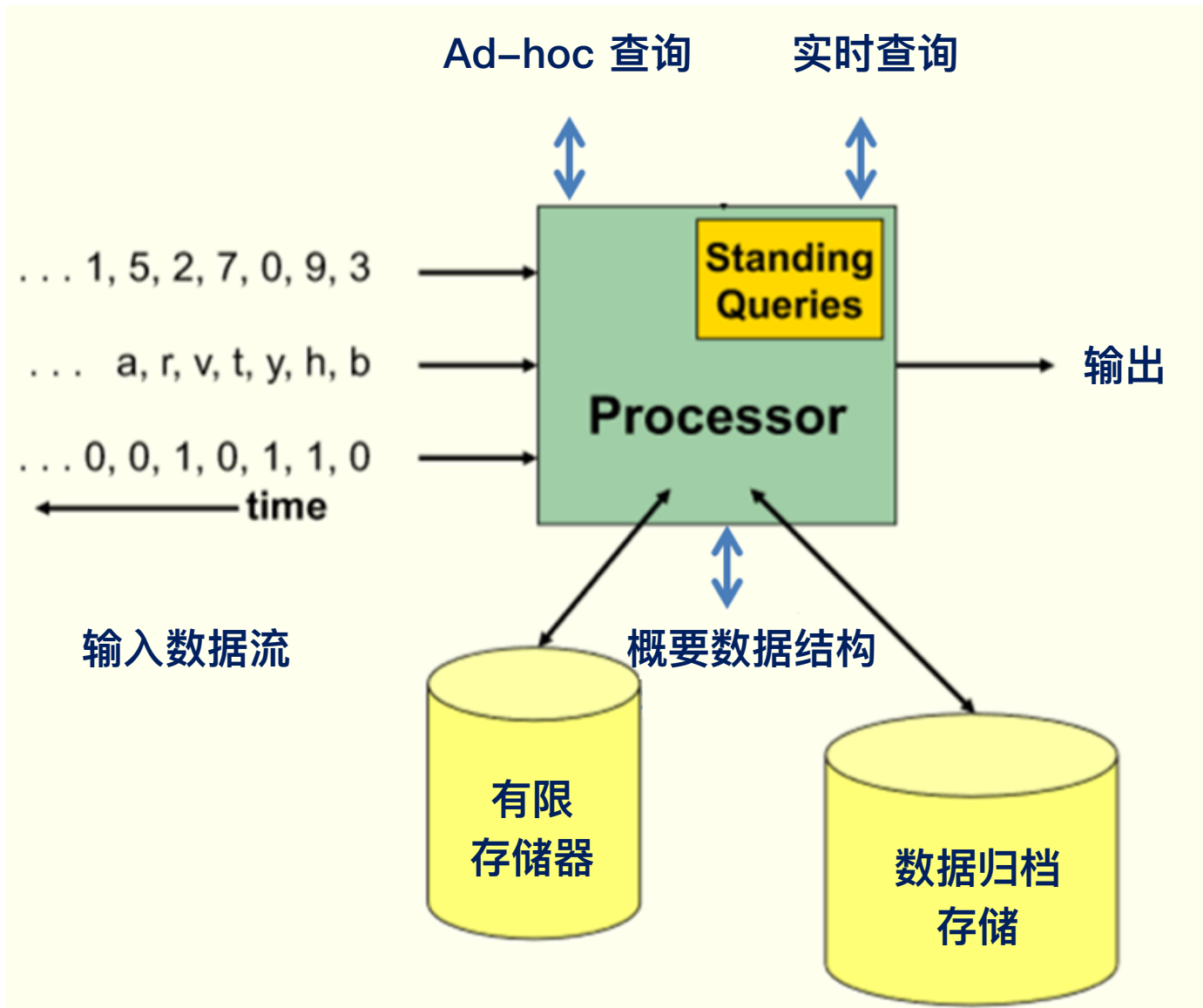


- 滑动窗口模型



- 衰减窗口模型

数据流分析挖掘框架



数据项频数估计

- 在数据流 $\sigma = \langle a_1, \dots, a_m, \dots \rangle, a_i \in [n]$ 中, 定义频数向量 $f = (f_1, \dots, f_n)$
- 其中 n 为 σ 中不同元素的个数, f_i 为元素 a_i 的频数且 $\sum_{i=1}^n f_i = m$
- 大多数问题: 如果 $\exists a_j : f_j > m/2$, 则输出 a_j , 否则输出 \emptyset
- 一般问题
 - 给定参数 k , 输出集合 $\{a_j : f_j \geq m/k\}$
 - 或给定参数 ψ , 输出集合 $\{a_j : f_j \geq \psi m\}$
- 例如: 给定数据流 $\sigma = \langle a, b, a, c, c, a, b, d \rangle$, 4 个不同元素的频数分别为 $f_a = 3, f_b = 2, f_c = 2, f_d = 1$
 - σ 中不存在大多数
 - 当 $k = 4$, 频繁项是 a, b, c
 - 当 $\psi = 0.3$, 频繁项是 a

课程提纲

Content

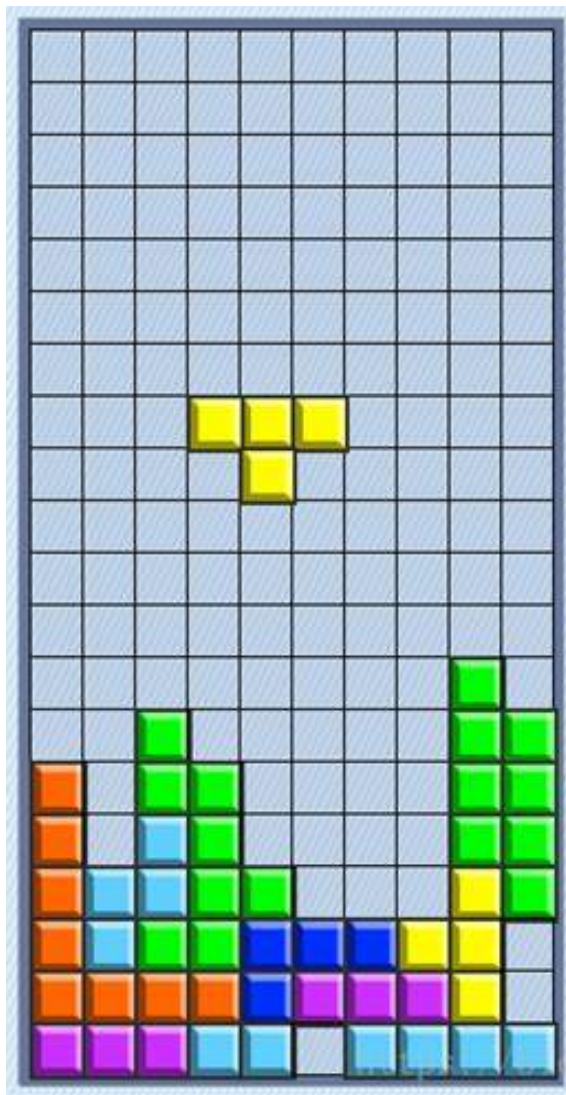
1 算法引入

2 Misra–Gries算法

3 Counting Sketch

4 Count–min Sketch

来自俄罗斯方块的启发



- 《俄罗斯方块》是一款由俄罗斯人阿列克谢·帕基特诺夫于1984年6月发明的休闲游戏
- 游戏规则
 - 小方块组成的不同形状陆续从屏幕上方落下来
 - 玩家通过调整对象的位置和方向，消去部分对象腾出空间
 - 没有被消除掉的方块不断堆积起来，直到游戏结束
- 类似于数据流

Misra–Gries 算法

输入: 数据流 $\sigma = \langle a_1, a_2, \dots, a_m \rangle$, $a_i \in [n]$, 正整数 k
输出: 数据流 σ 中的频繁元素集合 F

计数器: 俄罗斯方块游戏界面宽度

```
1  $F \leftarrow \emptyset$ ;  
2 while 数据流  $\sigma$  非空 do  
3   数据流  $\sigma$  中第  $i$  个元素到达;  
4   if  $a_i \in \text{keys}(F)$  then  
5      $F_i \leftarrow F_i + 1$ ;  
6   else  
7     if  $|\text{keys}(F)| < k - 1$  then  
8        $F_i \leftarrow 1$ ;  
9     else  
10      for  $a_j \in \text{keys}(F)$  do  
11         $F_j \leftarrow F_j - 1$ ;  
12        if  $F_j = 0$  then  
13          从  $F$  中移除  $a_j$ ;  
14 return 频数估计数组  $F$ ;
```

更新已有数据项的频数

一行未装满 (计数器未满),
新元素直接插入

一行装满 (计数器装满), 清除一行

Misra–Gries 算法示例

- 给定输入数据流 A, B, A, C, D, E, A, D , 且 $k = 3$

输入	操作	结果
----	----	----

--	--	--

Misra–Gries 算法分析 I

- **定理 1:** Misra–Gries 算法中的计数器减操作至多执行 $\lfloor \frac{m}{k} \rfloor$ 次, 其中 m 表示数据流中数据项个数, k 表示计数器个数
- **证明**
 - 每进行一次减操作计数器值之和比实际到达的数据项个数减少 k 个
 - 最终计数器值之和至少为 0, 且数据项的个数为 m
 - 所以, 减操作最多进行 $\lfloor \frac{m}{k} \rfloor$ 次

Misra–Gries 算法分析 II

- 定理 2: 当 $k = \lceil \frac{1}{\psi} \rceil$, 所有的 ψ -频繁项都会被 Misra–Gries 算法输出
- 证明
 - 由定理1可知, 计算器减操作最多执行了 $\lfloor \frac{m}{k} \rfloor$ 次
 - 因此算法结束时, 数据项的计数器值为 c_i , 满足 $c_i \leq f_i \leq c_i + \frac{m}{k}$
 - 对未被输出的数据项 i , 有 $c_i = 0$, 则 $f_i \leq \frac{m}{k} \leq \psi m$
 - 所以, 所有满足 $f_j > \psi m$ 的数据项 a_j , 即所有的 ψ -频繁项, 都会被 Misra–Gries 算法输出

Misra–Gries 算法分析 III

- Misra–Gries 算法是一个近似频繁项挖掘算法
- 优点
 - 逻辑简单
 - 高效算法
 - ✓ 空间复杂度为 $O(k)$, 与数据流规模无关
 - ✓ 处理每个数据项的复杂度为 $O(k)$
 - 所有频繁项一定会被输出
- 缺点
 - 可能会出现**误报**: 非频繁项也会被输出
 - 无法输出数据项的频数

课程提纲

Content

1 算法引入

2 Misra–Gries算法

3 **Count Sketch**

4 Count–min Sketch

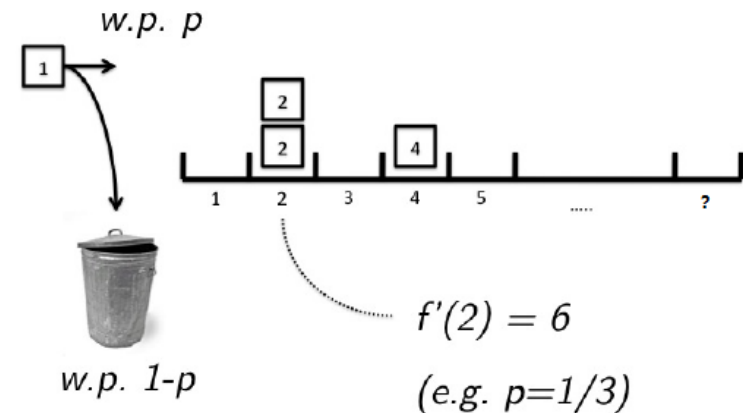
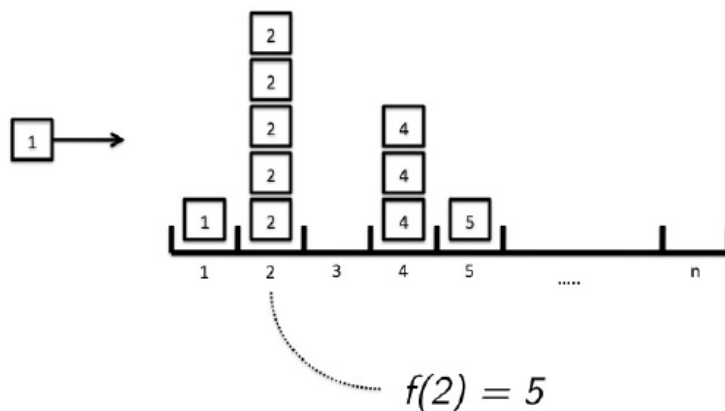
问题定义

- 频数查询

- 给定数据流 $\sigma = \langle a_1, \dots, a_m, \dots \rangle, a_i \in [n]$, 元素 i 的频数为 $f_i = |\{j : a_j = i\}|$ 。对于元素 i , 查询其频数 f_i

- 简单抽样算法

- 对每个到达的数据项, 以 $p = \frac{M}{m}$ 的概率对其频数加 1, 否则将其丢弃, 其中 M 表示抽样后的数据流大小



简单抽样方法分析

- 运用简单抽样方法, 元素 i 频数的 (ϵ, δ) -近似估计的空间复杂度是 $M = O\left(\frac{m \log(1/\delta)}{\epsilon^2}\right)$, 其中 m 为原数据流大小
- 证明:

定义随机变量 $X_{ij} = \begin{cases} 1, & \text{if } a_j = i \text{ is sampled} \\ 0, & \text{otherwise} \end{cases}$, 定义 $g_j = \sum_{i=1}^m X_{ij}$ 为简单

抽样后元素 i 的频数, 则元素 i 的频数估计值 $\hat{f}_i = \frac{g_i}{p}$

证明续

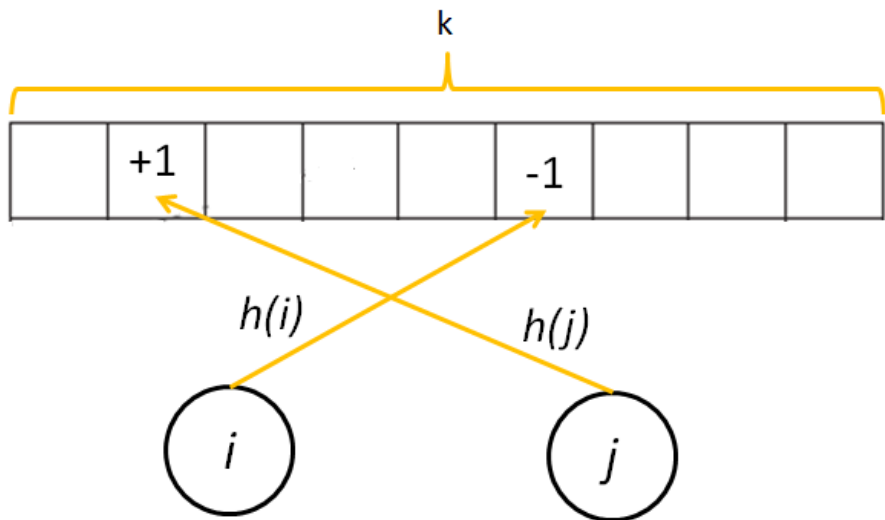
利用Chernoff不等式，得到

$$\begin{aligned} P(|\hat{f}_i - f_i| \geq \epsilon f_i) &= P(|g_i - pf_i| \geq \epsilon pf_i) \\ &= P(|\frac{g_i}{f_i} - p| \geq \epsilon p) \\ &< 2 \exp(-p\epsilon^2/3) < \delta \end{aligned}$$

将 $p = \frac{M}{m}$ 代入，即可得到： $M > \frac{3m \ln(2/\delta)}{\epsilon^2}$

- 简单抽样算法的空间复杂度为 $M = O(\frac{m \log(1/\delta)}{\epsilon^2})$
 - 时间复杂度与数据流大小有关
 - 无法确定抽样概率的大小（因为 m 可能未知）
 - 因此，简单抽样算法是无法应对快速到达的数据流

基本 Count Sketch



输入: 数据流, 查询元素 a

输出: 元素 a 出现的频数

- 1 初始化: $C[1 \dots k] \leftarrow 0$; $k = O(\frac{1}{\delta \epsilon^2})$;
- 2 选择哈希函数 $h : [n] \rightarrow [k]$
- 3 选择哈希函数 $g : [n] \rightarrow \{-1, 1\}$
- 4 处理: (j, c) , 其中 $c = 1$;
- 5 **while** 有数据到达 **do**
- 6 $C[h(j)] \leftarrow C[h(j)] + cg(j)$;
- 7 **return** $\hat{f}_a = g(a)C[h(a)]$;

哈希表的大小

哈希表中的频数更新

- 两个元素被 hash 到同一个位置

- 产生哈希冲突
- 但发生碰撞的元素频率可能增加, 也可能减少
- 由于增加或减少是完全随机的, 所以从期望来看是没有影响的

基本 Count Sketch 示例

- 假设输入数据流为 $\langle a, b, c, a, b, a \rangle$, 计数器个数为 2, $h(a) = h(b) \neq h(c)$, 试计算各元素的频数。
- 三个元素 a, b, c , 哈希函数 g 使得 $\{a, b, c\} \rightarrow \{-1, 1\}$ 有如下 8 种可能:

$g(a)$	$g(b)$	$g(c)$	$h(a) \text{ or } h(b)$	$h(c)$
+	+	+	$+3 + 2 = 5$	1
+	+	-	$+3 + 2 = 5$	-1
+	-	+	$+3 - 2 = 1$	1
+	-	-	$+3 - 2 = 1$	-1
-	+	+	$-3 + 2 = -1$	1
-	+	-	$-3 + 2 = -1$	-1
-	-	+	$-3 - 2 = -5$	1
-	-	-	$-3 - 2 = -5$	-1

$$f_a = \frac{(5 + 5 + 1 + 1) - (-5 - 5 - 1 - 1)}{8} = 3$$

$$f_b = \frac{(5 + 5 - 1 - 1) - (-5 - 5 + 1 + 1)}{8} = 2$$

$$f_c = \frac{(1 + 1 + 1 + 1) - (-1 - 1 - 1 - 1)}{8} = 1$$

基本 Count Sketch 分析

- 固定元素 a ，对每个元素 $j \in [n]$
 - 定义随机变量 $Y_j = \begin{cases} 1, & \text{if } h(j) = h(a) \\ 0, & \text{otherwise} \end{cases}$
 - Y_j 表示元素 j 和元素 a 是否被哈希到同一位置
 - $Y_j = 1$ 表示 $h(j) = h(a)$ ，即元素 j 对计数器 $C[h(a)]$ 有贡献
- 所有哈希到 $h(a)$ 位置上的元素都对 a 的频数估计都起到了作用，因此

$$\begin{aligned}\widehat{f}_a &= g(a)C[h(a)] = g(a) \sum_{j=1}^n f_j \cdot g(j) \cdot Y_j \\ &= f(a) + g(a) \sum_{j \in [n] \setminus \{a\}} f_j \cdot g(j) \cdot Y_j\end{aligned}$$

基本 Count Sketch 分析 (续)

- 对频数估计取期望 $E[\widehat{f}_a] = f(a) + g(a) \sum_{j \in [n] \setminus \{a\}} f_j \cdot E[g(j) \cdot Y_j]$
- 由于哈希函数 g 和 h 是独立的, 所以
$$E[g(j) \cdot Y_j] = E[g(j)] \cdot E[Y_j] = 0 \cdot E[Y_j] = 0$$
- 最终得到 $E[\widehat{f}_a] = f_a$
- 因此, 基本 Count Sketch 算法输出结果是对实际频数的无偏估计

基本 Count Sketch 分析 (续)

- 无偏估计 \neq 好的估计
- 算法输出值可能偏离真实值太大，所以仍需分析算法输出结果的方差
- 下面我们计算基本 Count Sketch 输出结果的方差

$$\begin{aligned}\text{Var}[\widehat{f_a}] &= \text{Var}\left[f(a) + g(a) \sum_{j \in [n] \setminus \{a\}} f_j \cdot g(j) \cdot Y_j\right] = g^2(a) \text{Var}\left[\sum_{j \in [n] \setminus \{a\}} f_j \cdot g(j) \cdot Y_j\right] \\&= g^2(a) E\left[\sum_{j \in [n] \setminus \{a\}} f_j \cdot g(j) \cdot Y_j\right]^2 - g^2(a) \left(E\left[\sum_{j \in [n] \setminus \{a\}} f_j \cdot g(j) \cdot Y_j\right]\right)^2 \\&= E\left[\sum_{j \in [n] \setminus \{a\}} f_j \cdot g(j) \cdot Y_j\right]^2 \\&= E\left[\sum_{j \in [n] \setminus \{a\}} f_j^2 Y_j^2 + 2 \sum_{i, j \in [n] \setminus \{a\} \wedge i \neq j} f_i f_j g(i) g(j) Y_i Y_j\right]\end{aligned}$$

基本 Count Sketch 方差 (续)

对 $j \in [n] \setminus \{a\}$, 可以得到 $E[Y_j^2] = E[Y_j] = P[h(j) = h(a)] = \frac{1}{k}$

对 $i \neq j \in [n] \setminus \{a\}$, 可以得到 $E[g(i)g(j)Y_iY_j] = E[g(i)]E[g(j)]E[Y_iY_j] = 0$

所以, $\text{Var}[\widehat{f}_a]$ 可以计算为

$$\begin{aligned}\text{Var}[\widehat{f}_a] &= E\left[\sum_{j \in [n] \setminus \{a\}} f_j^2 Y_j^2 + 2 \sum_{i, j \in [n] \setminus \{a\} \wedge i \neq j} f_i f_j g(i) g(j) Y_i Y_j\right] \\ &= \sum_{j \in [n] \setminus \{a\}} \frac{f_j^2}{k} + 0 = \frac{\|f\|_2^2 - f_a^2}{k} := \frac{\|f_{-a}\|_2^2}{k}\end{aligned}$$

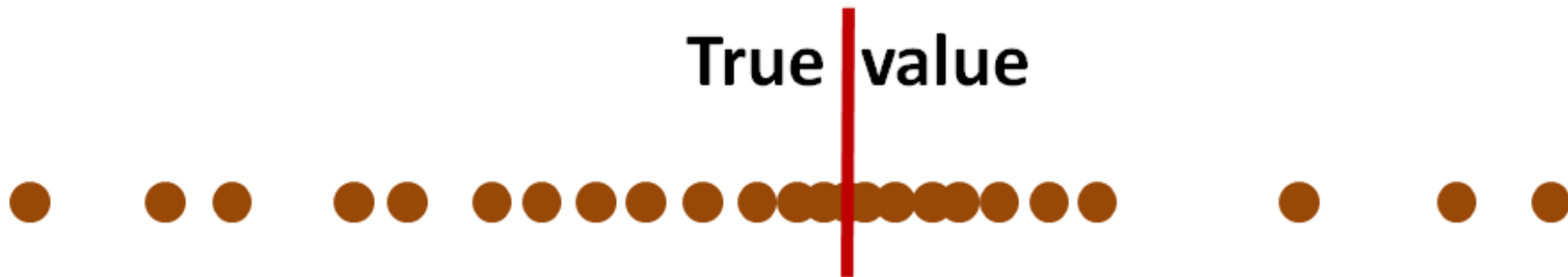
因此, 随 k 值的增大, 即存储计数数组的空间增大, 方差随之减小

Count Sketch 算法

- 根据 Chebyshev 不等式, 当 $k = O(\frac{1}{\delta\epsilon^2})$ 时, 可以得到 f_a 的 (ϵ, δ) -近似估计

$$P[|\hat{f}_a - f_a| \geq \epsilon \|f\|_2] \leq P[|\hat{f}_a - f_a| \geq \epsilon \|f_a\|_2] \leq \frac{\text{Var}[\hat{f}_a]}{\epsilon^2 \|f_a\|_2^2} = \frac{1}{k\epsilon^2} < \delta$$

- 运用 Tug of War 技术, 改进算法



Count Sketch 算法

输入: 数据流, 查询元素 a

输出: 元素 a 出现的频数

比基本 Count Sketch 中的计数器数量要少

1 初始化: $C[1 \cdots k] \leftarrow 0, k = \frac{3}{\epsilon^2}, t = O(\log(1/\delta));$

2 选择 t 个哈希函数 $h_1, \dots, h_t : [n] \rightarrow [k]$

3 选择 t 个哈希函数 $g_1, \dots, g_t : [n] \rightarrow \{-1, 1\}$

4 处理: (j, c) , 其中 $c = 1$;

5 **while** 有数据到达 **do**

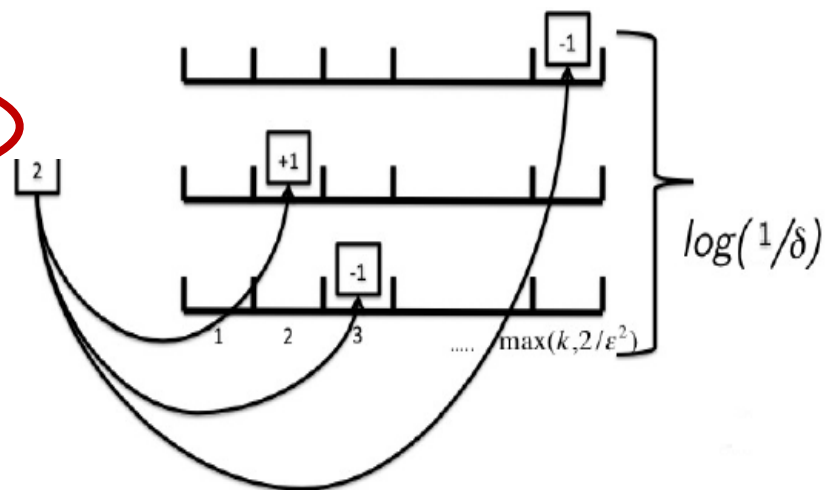
6 **for** $i = 1$ **to** t **do**

7 $C[i][h_i(j)] \leftarrow C[i][h_i(j)] + cg_i(j);$

8 **return** $\hat{f}_a = \text{median}_{1 \leq i \leq t} g_i(a) C[i][h_i(a)];$

虽然单次可能不准, 但是
Count Sketch 做了多次

输出多次基本 Count Sketch
输出结果的中位数



Count Sketch 算法分析

- 定义 $Y_i = \begin{cases} 1, & \text{if } |\hat{f}_a - f_a| \geq \epsilon \|f\|_2 \\ 0, & \text{otherwise} \end{cases}$
- 适当选择 $k = O(1/\epsilon^2)$, 使得 $P(Y_i = 1) < 1/3$
- 注意 $\mu = E(\sum_i Y_i) < t/3$, 由 Chernoff 不等式可得
$$P(\sum_i Y_i > t/2) \leq P(\sum_i Y_i > (1 + 1/2)\mu) \leq \exp(-\frac{\mu}{12})$$
$$\exp(-\frac{t}{36}) \leq \exp(-\frac{\mu}{12}) < \delta, \text{ 即 } t = O(\log(1/\delta))$$
- 最终得到一个空间消耗为 $O(\frac{\log(1/\delta)}{\epsilon^2})$ 的 (ϵ, δ) -近似估计

课程提纲

Content

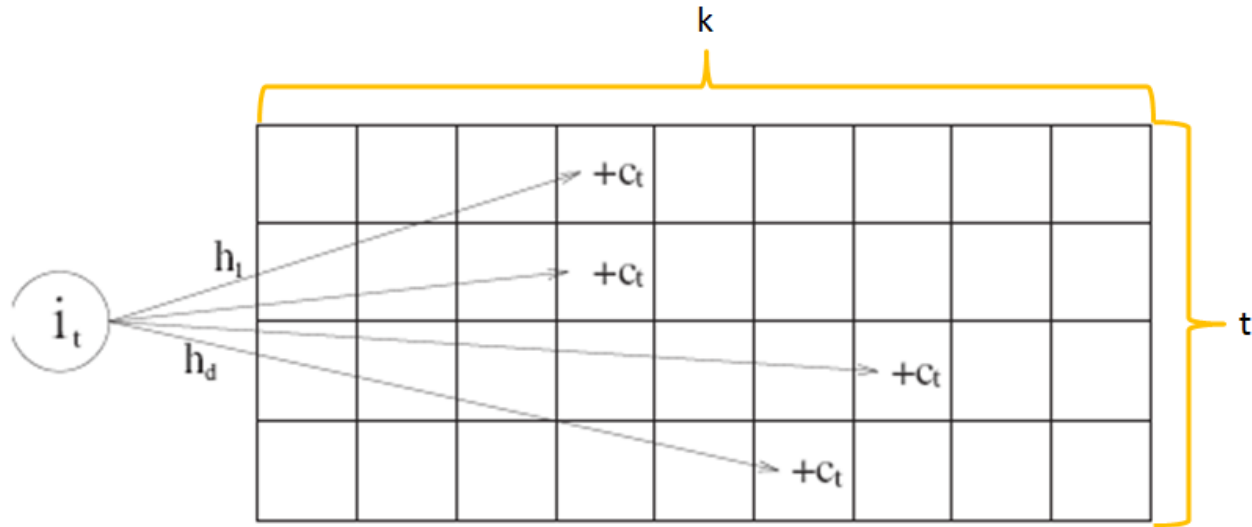
1 算法引入

2 Misra–Gries算法

3 Counting Sketch

4 Count–min Sketch

Count-min Sketch



输入: 数据流, 查询元素 a

输出: 元素 a 出现的频数

- 1 初始化: $C[1 \cdots d][1 \cdots w] \leftarrow 0, w = \frac{2}{\epsilon}, d = \lceil \log(1/\delta) \rceil$; 选择 d 个独立的哈希函数 $h_1, h_2, \dots, h_d : [n] \rightarrow [w]$
- 2 处理: (j, c) , 其中 $c = 1$;
- 3 for $i = 1$ to d do
- 4 $C[i][h_i(j)] \leftarrow C[i][h_i(j)] + c$;
- 5 return $\hat{f}_a = \min_{1 \leq i \leq d} C[i][h_i(a)]$;

哈希表的大小

哈希函数的个数

返回 d 个哈希表中最小的值

CM Sketch 分析

- 对元素 $j \in [n] \setminus \{a\}$, 定义 $Y_{i,j} = \begin{cases} 1, & \text{if } h_i(j) = h_i(a) \\ 0, & \text{otherwise} \end{cases}$
- 当 $Y_{i,j} = 1$ 时, 元素 j 在第 i 个哈希函数上与元素 a 发生了哈希冲突
- 给定元素 a 和哈希函数 h_i , 定义随机变量 X_i 为第 i 个哈希函数上其他元素对元素 a 频数估计值的贡献, 大小为 $X_i = \sum_{j \in [n] \setminus \{a\}} f_j Y_{i,j}$
- 根据期望的线性性质以及 $E[Y_{i,j}] = \frac{1}{k}$, 我们得到

$$E[X_i] = \sum_{j \in [n] \setminus \{a\}} \frac{f_j}{k} = \frac{\|f\|_1 - f_a}{k} := \frac{\|f_{-a}\|_1}{k}$$

CM Sketch 分析

- 因为 $f_j \geq 0$, 所以 $X_j \geq 0$, 根据马尔科夫不等式, 选择合适的 k 值, 得到不等式 $P[X_i \geq \epsilon \|f\|_1] \leq P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{\|f_{-a}\|_1}{k\epsilon \|f_{-a}\|_1} = \frac{1}{2}$
- 对 d 个相互独立的哈希函数, 若 $\widehat{f}_a - f_a \geq x$, 则 $\min\{X_1, \dots, X_d\} \geq x$, 即 $P[\widehat{f}_a - f_a \geq \epsilon \|f_{-a}\|_1] = P[\min\{X_1, \dots, X_d\} \geq \epsilon \|f_{-a}\|_1] = \prod_{i=1}^d P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{1}{2^d}$
- 可以选择适当 d 的值, 使得上式概率上界至多为 δ , 所以至少以 $(1 - \delta)$ 的概率下式成立: $f_a \leq \widehat{f}_a \leq f_a + \epsilon \|f_{-a}\|_1$
- 该算法需要的计数器个数为 $M = O\left(\frac{\log(1/\delta)}{\epsilon}\right)$

本章小结

- 频繁项挖掘
 - Misra–Gries 算法
 - Count sketch 算法
 - Count–min sketch 算法
- 数据流场景下的查询与分析
 - 数据高速达到、容量不限，查询需要实时响应
 - 对算法空间和时间复杂度的要求很高
 - 通常设计小巧的 sketch 以达到目的
 - 而且需要深入的算法分析，保证算法精度
- 除了频繁项挖掘，还有其他的一些任务
 - 不同元素个数
 - Top–k 查询
 - 聚类
 -