频率的凹次线性函数的抽样草图

Edith Cohen, Ofir Geri

摘要

我们考虑由建模为键值对的元素组成的大规模分布式数据集，以及计算统计数据或聚合的任务，其中每个键的贡献由其频率的函数 (其元素值的和) 加权。这个基本问题在数据分析和机器学习中有大量的应用，特别是用频率的凹次线性函数，可以减轻高频键的不均衡影响。凹次线性函数族包括低频矩 ($p \leq 1$)、capping、对数及其组合。一种常见的方法是对键进行抽样，理想情况下，按其贡献的比例进行抽样，并从抽样中估计统计数据。一种简单但昂贵的方法是聚合数据，生成键及其频率的表，将我们的函数应用于频率值，然后应用加权抽样方案。我们的主要贡献是设计了可组合的采样草图，可以定制为任何凹的次线性函数的频率。我们的草图结构大小非常接近于期望的样本大小，我们的样本对估计质量提供了统计保证，它非常接近于在聚合数据上计算的相同大小的理想样本的质量。最后，我们通过实验证明了我们的方法的简单性和有效性。

关键词：凹的次线性函数；采样草图

1 引言

当数据元素的数量非常大时，我们可以通过计算一个较小的随机样本来有效地估计聚合，这在实际工作中通常是有帮助的。在某些应用程序中，获取样本可能是最终目标。例如，当聚合是梯度时，我们可以使用样本本身作为一个随机梯度。为了在统计上保证我们的估计质量，我们需要进行加权抽样，较重 (频率较高) 的键具有较高的抽样概率。

典型的数据集具有倾斜的频率分布，其中一小部分键具有非常大的频率，我们可以通过抑制它们的影响获得更好的结果或学习更好的数据模型。其做法是对频率使用一个凹次线性函数 f，这样键的重要性权重变为 f(v_x)，而不是简单的频率 v_x。

假设草图 S(D) 是一个数据结构，它总结了一组数据元素 D，因此我们可以从草图 S(D) 中恢复 D 感兴趣的输出 (在我们的例子中，是键的样本)。如果我们能从两个元素集合的草图 $S(D_1)$ 和 $S(D_2)$ 中得到两个元素集合 D_1 和 D_2 的草图 $S(D_1 \cup D_2)$，那么一个草图结构是可组合的。这个特性本身就为并行化或分布计算提供了充分的灵活性。

2 相关工作

对于聚合数据集 (其中键对元素是唯一的)，有多种经典的可组合加权抽样方案。提供严格的最坏情况方差边界估计量的方案包括优先级 (顺序泊松) 抽样和 VarOpt 抽样。我们将重点放在 PPSWOR，并将其作为基础方案，因为它可以扩展到未聚合的数据集，其中多个元素可以增加每个键的频率/权重。

我们的草图所需的空间与需要聚合数据的方法相比有了显著的改进。如果期望的样本量是 k，我们表明，草图在任何给定时间所需的空间的期望是 O(k)。在处理数据元素 D 时，空间在任何时候都

不会超过 $O(k+min\{logm, loglog(\frac{Sum_D}{Min(D)})\}+log(\frac{1}{\delta}))$ 的概率至少为 $1-\delta$，其中 m 为 D 中的元素个数，Min(D) 为 D 中元素的最小值，Sum_D 为所有键的频率之和[1]。

2.1 未聚合数据集的组合抽样草图的研究

对于未聚合数据集的组合抽样草图的研究持续了几十年，目标是满足在聚合频率上计算的样本质量，同时使用只能容纳最终样本大小数量的不同键的草图结构。其中包括用于 distinct sampling(当 v>0 时，f(v)=1) 的 folklore 草图和用于 sum sampling(f(v)=v) 的草图结构。

2.2 基于随机线性投影的草图

ℓ_p 抽样草图根据 $f(v)=v^p$，$p\in[0,2]$ 粗略地抽样。与基于样本的草图相比，这些草图具有更高的开销，并且不支持频率的所有凹次线性函数。在某些方面，它们在应用程序中受到更多的限制——例如，它们不是设计用来生成包含原始键的样本的。它们的优点是可以用于频率的超线性函数一起使用，也可以支持带符号的元素值 (turnstile 模型)。

3 本文方法

3.1 本文方法概述

我们在这里考虑的数据集以未聚合的形式呈现: 每个键可以在不同的位置出现多次。本文复现工作的重点是实现论文中提到的可组合的草图结构，如 Bottom-k、PPSWOR、SumMax，允许计算关于权重 $f(v_x)$ 的未聚合数据的样本。其中 Bottom-k 结构用于维护 k 个数据元素: 对于每个键，只考虑具有该键的最小值的元素。在这些元素中，该结构保留了 k 个值最低的元素。PPSWOR 维护了一个可组合的 Bottom-k 结构，这样对于每个键 x，具有键 x 的元素 (用 seed(x) 表示) 的最小值独立于 Eep(v_x) 被抽出。SumMax 是一个辅助草图，用来处理元素 e=(e.key,e.val)，键 e.key=(e.key.p,e.key.s) 结构为具有一个主键 e.key.p 和一个从键 e.key.s。它的目标是根据权重 $SumMax_D$(x) 生成主键 x 的 PPSWOR 样本。

从未聚合数据中计算样本的一种方法是，首先聚合数据，生成键-频率对表 (x,v_x)，然后利用凹次线性函数计算权重 $f(v_x)$，并应用加权抽样方案。

3.2 常见的凹次线性函数

f(v)=ln(1+v)

f(v)=v^p

f(v)=min{T,v} ($T\geqslant 0$)

3.3 凹次线性函数的拉普拉斯逆变换

根据论文 HyperLogLog Hyper Extended: Sketches for Concave Sublinear Frequency Statistics，我们可以得出 f(v) = $ln(1+v)$ 的拉普拉斯逆变换为 a(t) = $\frac{e^{-t}}{t}$, f(v) = v^p 的拉普拉斯逆变换为 a(t) = $\frac{p}{\Gamma(1-p)}t^{-(1+p)}$。

再根据这篇论文，利用上面得到的拉普拉斯逆变换，当 f(v)=ln(1+v) 时，$A(\gamma)=\int_{\gamma}^{\infty} a(t)dt=\int_{\gamma}^{\infty}\frac{e^{-t}}{t}dt$，$B(\gamma)=\int_{0}^{\gamma} ta(t)dt=1-e^{-\gamma}$。当 f(v)=$v^{0.5}$ 时，$A(\gamma)=\int_{\gamma}^{\infty} a(t)dt=\frac{1}{\sqrt{\pi\gamma}}$，$B(\gamma)=\int_{0}^{\gamma} ta(t)dt=\sqrt{\frac{\gamma}{\pi}}$

4 复现细节

4.1 与已有开源代码对比

在最后利用样本草图计算 $f(v_x)$ 的逆概率估计量时，引用了作者发布的计算条件包含概率的代码，如图 1 所示。

```
def seedCDF(w, t, gamma, r, funcA, funcB):
    p1 = numpy.exp(-1.0 * w * t * funcB(gamma))
    p2_low = (numpy.exp(-1.0 * funcA(gamma) * t / r) * (1.0 - numpy.exp(-1.0 * w * gamma)))
    func_to_integr = lambda x: w * numpy.exp(-1.0 * w * x - (t * funcA(x) / r))
    p2_high = integrate.quad(func_to_integr, gamma, numpy.inf)
    p2 = p2_low + p2_high[0]
    return 1.0 - p1 * (p2 ** r)
```

图 1: 引用代码

4.2 伪代码

Algorithm 1 Bottom-k Sketch Structure

// Initialize structure
Input: the structure size k
s.set ← ∅ // Set of ⩽ key-value pairs
// Process element
Input: element *e=(e.key, e.val),* a bottom-k structure s
if *e.key* ∈ $s.set$ **then**
  replace the current value v of e.key in s.set with min{v,e.val}
else
  insert(e.key,e.val) to s.set
  if *|s.set| = k+1* **then**
    Remove the element e' with maximum value from s.set
  end
end
// Merge two bottom-k structures
Input: s_1, s_2 // Bottom-k structures
Output: s // Bottom-k structure
$P \leftarrow s_1.set \cup s_2.set$
$s.set \leftarrow$ *the (at most) k elements of P with lowest values (at most one element per key)*

Algorithm 2 PPSWOR Sampling Sketch

// Initialize structure
Input: the sample size k
Initialize a bottom-k structure s.**sample** // Algorithm 1
// Process element
Input: element *e=(e.key, e.val),* PPSWOR sample structure s
v ∼ Exp[e.val]
Process the element (e.key, v) into the bottom-k structure s.sample
// Merge two structures $s_1, s_2 toobtains$
$s.sample \leftarrow$ Merge the bottom-k structures $s_1.sample\ and\ s_2.sample$

Algorithm 3 SumMax Sampling Sketch

// Initialize empty structure s
Input: Sample size *k*
$s.h \leftarrow$ fully independent random hash with range Exp[1]
Initialize s.sample // A bottom-k structure (Algorithm 1)
// Process element e = (e.key, e.val) where e.key = (e.key.p, e.key.s)
Input: element *e=(e.key, e.val),* PPSWOR sample structure *s*
v $\sim$ Exp[e.val]
Process the element (e.key, v) into the bottom-k structure s.**sample**
Process element (e.key.p, s.h(e.key)/e.val) to structure s.sample // bottom-k process element (Algorithm 1)
// Merge structures $s_1, s_2 (with s_1.h = s_2.h) to get s$
$s.h \leftarrow s_1.h$
$s.sample \leftarrow$ Merge $s_1.sample, s_2.sample$ // $bottom-k\ merge\ (Algorithm\ 1)$

Algorithm 4 Produce a Final Sample from a Sampling Sketch Structure

Input: Sampling sketch structure *s* for *f*
Output: Sample of size *k* of key and seed pairs
if $\int_\gamma^\infty a(t)dt > 0$ **then**
  foreach *e* $\in$ *s.Sideline* **do**
    Process element (e.key, $\int_\gamma^\infty a(t)dt$) by sketch s.SumMax
  end
end
foreach *e* $\in$ *s.SumMax.sample* **do**
  $e.val \leftarrow$ r*e.val // Multiply value by r
end
if $\int_0^\gamma ta(t)dt > 0$ **then**
  foreach *e* $\in$ *s.ppswor.sample* **do**
    e.val $\leftarrow \frac{e.val}{\int_0^\gamma a(t)dt}$ // Divide value by B(γ)
  end
  sample $\leftarrow$ merge s.SumMax.sample and s.ppswor.sample // Bottom-k merge (Algorithm 1)
else
  sample $\leftarrow$ s.SumMax.sample
end
return sample

Algorithm 5 Sampling Sketch Structure for f

// Initialize empty structure s
Input: k: Sample size, ε, $a(t) \geqslant 0$
Initialize s.SumMax // SumMax sketch of size k (Algorithm 3)
Initialize s.ppswor // PPSWOR sketch of size k (Algorithm 2)
Initialize $s.sum \leftarrow 0$ // A sum of all the elements seen so far
Initialize $s.\gamma \leftarrow \infty$ // Threshold
Initialize s.Sideline // A composable max-heap/priority queue
// Process element
Input: Element *e = (e.key,e.val),* structure *s*
Process e by s.ppswor
$s.sum \leftarrow$ s.sum + e.val
$s.\gamma \leftarrow \dfrac{e.val}{s.sum}$
// r = k/ε
foreach $i \in [r]$ **do**
  $y \sim Exp[e.val]$ // *Exponentially distributed with parameter e.val*
  // *Process in Sideline*
  if *The key(e.key,i) appears in s.Sideline* **then**
    Update the value of (e.key, i) to be the minimum of y and the current value
  else
    Add the element ((e.key, i), y) to s.Sideline
  end
end
while *s.Sideline contains an element g = (g.key, g.val) with g.val* $\geqslant s.\gamma$ **do**
  Remove g from s.Sideline
  if $\int_{g.val}^{\infty} a(t)dt > 0$ **then**
    Process element (g.key, $\int_{g.val}^{\infty} a(t)dt$*) by s.SumMax*
  end
end
// Merge two structures *s1 and s2 to s (with same k,* ε, a *and same* h *in* $SumMax\ sub-structures$)
$s.sum \leftarrow s_1.sum + s_2.sum$
$s.\gamma \leftarrow \dfrac{e.val}{s.sum}$
$s.Sideline \leftarrow$ *merge* $s_1.Sideline and s_2.Sideline // Merge priority queues.$
$s.ppswor \leftarrow$ *merge* $s_1.ppswor and s_2.ppswor // Merge PPSWOR structures$
$s.SumMax \leftarrow$ *merge* $s_1.SumMax and s_2.SumMax // Merge Sun Max structures$
while *s.Sideline contains an element g = (g.key, g.val) with g.val* $\geqslant s.\gamma$ **do**
  Remove g from s.Sideline
  if $\int_{g.val}^{\infty} a(t)dt > 0$ **then**
    Process element (g.key, $\int_{g.val}^{\infty} a(t)dt$*) by s.SumMax*
  end
end

4.3 使用说明

我们使用的数据集为 abcnews(澳大利亚广播公司发布的新闻标题)。首先读取数据集获得所有的新闻标题，将每个标题按空格划分成若干个单词，对于每一个单词，将其值定为 1，创建一个以单词作为 key，value 为 1 的 (key，value) 键值对。然后将所有的键值对放入一个列表中，生成一个 elements 列表。最后设置草图的抽样大小 k，参数 ε(用于影响处理一个元素的运行时间)，将 elements，k，ε 传给采样算法，生成采样草图，等草图生成之后利用该草图估算数据集的统计量。

5 实验结果分析

样本草图大小 k=25，参数 ε=0.5，凹次线性函数 f(v)=ln(1+v)。抽样结果如图 2 所示，估算结果如图 3 所示。

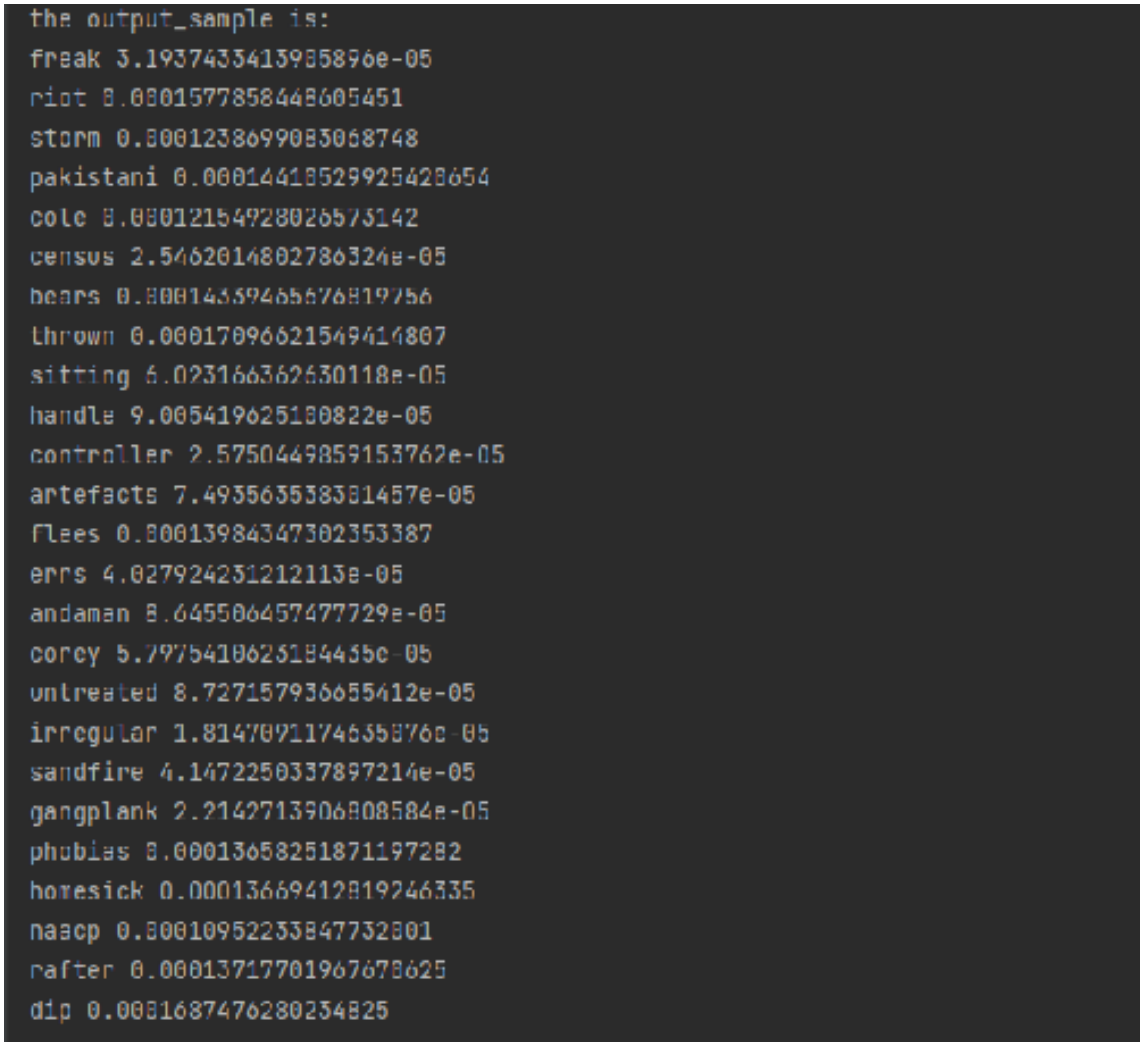

图 2: 样本草图

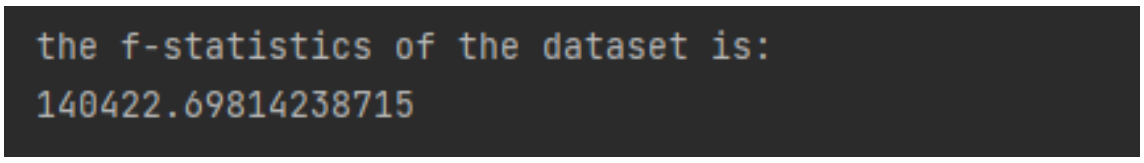

图 3: 估算结果

6 总结与展望

根据论文中的伪代码，查阅论文中给出的相关参考文献，引用作者发布的部分代码，对论文进行了一个简单的实现，可以做到对大规模数据集抽取数据，构建自定义大小的抽样草图，然后利用草图对整个数据集进行估计。

但是在整个实现过程还存在一些不足。一是在最后估算时，未能写出计算条件包含概率的代码，而是直接引用的作者发布的代码，二是在生成最终的抽样草图时，未能实现合并两个抽样草图的算法，这样当数据是分布式存储在多个集群上时，就不能生成最终的抽样草图。对于这一点，在未来还需要进一步改进。

参考文献

[1] COHEN E, GERI O. Sampling Sketches for Concave Sublinear Functions of Frequencies[J]. Neural Information Processing Systems, 2019: 1361-1371.