Sampling Sketches for Concave Sublinear Functions of Frequencies

Edith Cohen[1] and Ofir Geri[2]

[1]Google Research, CA and Tel Aviv University, Israel

[2]Stanford University, CA

Motivation

- We are given a distributed or streaming set of elements

  … `"hike"` `"book"` … `"news"` `"hike"` …

  (search queries, users, terms, videos…)
- Denote the frequency of key x by w_x
- We wish to evaluate a function for different parameters θ:

$$\ell(\mathcal{X};\theta) = \sum_{x\in\mathcal{X}} f(w_x)L(x;\theta)$$

- The function f is concave sublinear
  - Used to mitigate the disproportionate effect of large keys
- Application: computing word embeddings
  - word2vec: $f(x) = x^{0.5}$ and $f(x) = x^{0.75}$
  - GloVe: $f(x) = \min\{x^{0.75}, T\}$

Problem Definition

- Distributed/streaming data: elements of the form (key, value)
- The data is unaggregated: the same key may appear multiple times
- The frequency w_x of key x is the sum of values of elements with key x
- Goal: compute statistics that are linear in $f(w_x)$, for example, $\ell(\mathcal{X};\theta) = \sum_{x\in\mathcal{X}} f(w_x)L(x;\theta)$

- Naive solution: compute a table
- Too expensive when the number of keys is large

x	w_x	$f(w_x)$
…	…	…
…	…	…

- Common approach: produce a sample of k keys
  - Estimate the statistics from the sample
  - Goal is to optimize estimate quality (minimize variance)

- Scheme for unaggregated data: PPSWOR [Rosén '97]
  - Optimal when $f(w_x) = w_x$
- This work: can we extend to other functions f (without computing the table)?

Our Contribution

- We design composable sampling sketches tailored to any concave sublinear f
- The estimate quality (variance) is almost optimal
- The expected space at any specific time $O(k)$
  - Additionally show worst-case bounds on the maximum space used (at all times)
- Requires two passes: one to pick the sample, the second to compute estimators

Concave Sublinear Functions

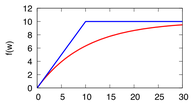
- f is concave sublinear if it is a non-negative combination of cap functions

$$cap_T(w) = \min\{w, T\}$$

- For every T, $T(1-e^{-w/T})$ approximates $\min\{w,T\}$, and this approximation leads to the following definition
- f is soft concave sublinear if for some $a(t) \geq 0$,

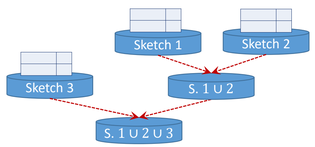
$$f(w) = \int_0^\infty a(t)(1-e^{-wt})dt$$

- Examples for (approximately) soft concave sublinear:
  $f(w) = w^p$ (for $p \leq 1$)
  $f(w) = \ln(1+w)$
  $f(w) = \min\{w^{0.75}, T\}$ (for a constant $T > 0$)



Composable Sketches

- Given the sketches $S(D_1)$ and $S(D_2)$ of two datasets D_1 and D_2 (respectively), we can compute the sketch $S(D_1 \cup D_2)$ of the dataset $D_1 \cup D_2$
- This gives full flexibility to distribute/parallelize the computation



- Composable sketches can also be used to process streaming data

Estimation using a Sample

- Given a sample S, we use the inverse-probability estimator $\widehat{w_x}$ for the frequency of key x [Horvitz, Thompson '52]:

$$\widehat{w_x} = \begin{cases} \frac{w_x}{\Pr[x\in S]} & x \in S \\ 0 & x \notin S \end{cases}$$

- The estimator is unbiased $E[\widehat{w_x}] = w_x$
- Can similarly define $\widehat{f(w_x)}$
- To estimate statistics over the entire dataset, enough to sum over S:

$$E\left[\sum_{x\in S}\widehat{w_x}\right] = E\left[\sum_{x\in\mathcal{X}}\widehat{w_x}\right] = \sum_{x\in\mathcal{X}} w_x$$

PPSWOR

- Baseline sampling scheme: probability proportional to size and without replacement sampling [Rosén '97]
- To get a sample of size k, perform k steps:
  At each step, pick without replacement one key, such that x is chosen with probability proportional to w_x
- A "gold standard" sample: tight optimal worst-case bounds on the variance

- Our goal in this work is to produce a sample of k keys and unbiased estimators $\widehat{f(w_x)}$
- Ideally, we would like to perform PPSWOR sampling according to $f(w_x)$

Main Result and Sketch Overview

- Our sampling sketch computes a sample and estimators $\widehat{f(w_x)}$ that are almost optimal
  - The variance is at most 5 times the variance bound for PPSWOR

- Any soft concave sublinear f can be represented as $f(w) = \int_0^\infty a(t)(1-e^{-wt})dt$
- For threshold γ, we store separate samples for
  $f_{low}(w) = \int_0^\gamma a(t)(1-e^{-wt})dt$
  $f_{high}(w) = \int_\gamma^\infty a(t)(1-e^{-wt})dt$
- If we pick γ right, f_{low} needs just a PPSWOR sketch
  - Simple argument using $x - \frac{x^2}{2} \leq 1 - e^{-x} \leq x$
- f_{high} builds on an earlier framework and two novel components: the analysis of stochastic PPSWOR sampling and the SumMax sampling sketch

- Earlier framework [Cohen '17]: replace each element with an "output" element

key	value	⟶	key	New random value

  Let $Max(x)$ be the maximum value of output element with key x. Then,

$$E[Max(x)] = f_{high}(w_x)$$

- We see the random $Max(x)$ and not its expected value.
  Is $Max(x)$ close enough to $E[Max(x)]$?

  Component 1: Stochastic PPSWOR Sampling
  - Shows that under certain assumptions, a PPSWOR sample on random weights w_x is similar to a PPSWOR sample with weights $E[w_x]$
  - For worst case guarantees, requires generating $2k$ temporary output elements per input element
- How do we sample according to the average of the $2k$ copies of $Max(x)$?
  Component 2: SumMax Sampling Sketch
  - Takes output elements with key (x, i)
  - Returns a PPSWOR sample of keys using weights $\frac{1}{2k}\sum_{i=1}^{2k} Max(x,i)$

Data

x	value

⟶ [Cohen '17]

$(x,1)$	random value
⋮	
$(x,2k)$	random value

⟶ SumMax

Sample

x	

Experiments

- Simple Python 2.7 implementation
- Practical optimizations: removing redundant keys from substructures
- We estimate the f-statistics $\sum_x f(w_x)$
  - For $f(w) = w^{0.5}$ and $f(w) = \ln(1+w)$
  - Using samples of size $k \in \{25,50,75,100\}$
- Benchmarks: PPSWOR and priority sampling on aggregated data
- Datasets:
  - abcnews [Kulkarni '17]: News headlines. For each word, an element with value 1.
  - flicker [Plangprasopchok, Lerman, Getoor '10]: Tags used by Flicker users to annotate images. The key of each element is a tag, and the value is the number of times it appeared in a certain folder.
  - Three synthetic generated datasets, where each element has value 1 and the key is drawn from the Zipf distribution with parameter α, for $\alpha \in \{1.1, 1.2, 1.5\}$

- The error is significantly lower than the worst-case bound and is close to the error achieved by the benchmarks
- The number of distinct keys stored at any time is close to k, and the total sketch size (number of elements stored) rarely exceeded $3k$

k	NRMSE		Benchmarks		max #keys		max #elem	
	bound	actual	PPSWOR	Pri.	ave	max	ave	Max
$f(w) = w^{0.5}$, 200 repetitions								
Dataset: abcnews (7.07 × 10^6 elements, 91.7 × 10^3 keys)								
25	0.834	0.213	0.213	0.217	31.7	37	50.9	76
50	0.577	0.142	0.128	0.137	58.5	66	95.1	136
75	0.468	0.120	0.111	0.110	85.4	94	134.8	181
100	0.404	0.105	0.098	0.103	111.2	120	171.1	256
Dataset: flickr (7.64 × 10^6 elements, 572.4 × 10^3 keys)								
25	0.834	0.200	0.190	0.208	31.2	37	53.1	77
50	0.577	0.144	0.147	0.142	57.8	64	94.6	130
75	0.468	0.123	0.114	0.110	83.7	91	131.7	175
100	0.404	0.115	0.095	0.099	108.9	116	173.4	223
Dataset: zipf1.1 (2.00 × 10^6 elements, 652.2 × 10^3 keys)								
25	0.834	0.215	0.198	0.217	31.8	39	52.5	75
50	0.577	0.123	0.137	0.131	58.7	66	95.0	130
75	0.468	0.109	0.115	0.114	84.7	91	135.2	186
100	0.404	0.106	0.103	0.097	111.2	119	176.3	221
Dataset: zipf1.2 (2.00 × 10^6 elements, 237.3 × 10^3 keys)								
25	0.834	0.199	0.208	0.214	31.1	38	53.2	83
50	0.577	0.144	0.138	0.145	57.9	65	98.4	139
75	0.468	0.122	0.116	0.124	83.9	90	138.2	173
100	0.404	0.098	0.109	0.096	109.6	115	179.2	227
Dataset: zipf1.5 (2.00 × 10^6 elements, 22.3 × 10^3 keys)								
25	0.834	0.201	0.207	0.194	30.1	35	53.4	74
50	0.577	0.152	0.139	0.142	56.1	60	101.5	136
75	0.468	0.115	0.115	0.112	81.6	86	151.8	199
100	0.404	0.098	0.094	0.086	107.1	113	196.3	248
$f(w) = \ln(1+w)$, 200 repetitions								
Dataset: abcnews (7.07 × 10^6 elements, 91.7 × 10^3 keys)								
25	0.834	0.208	0.217	0.194	29.5	34	49.1	71
50	0.577	0.138	0.136	0.142	54.9	60	80.9	110
75	0.468	0.130	0.099	0.117	80.0	85	111.1	152
100	0.404	0.102	0.115	0.103	104.9	109	140.7	184
Dataset: flickr (7.64 × 10^6 elements, 572.4 × 10^3 keys)								
25	0.834	0.227	0.199	0.180	28.0	31	41.4	69
50	0.577	0.144	0.151	0.129	53.3	59	72.2	101
75	0.468	0.119	0.121	0.109	78.2	83	99.8	135
100	0.404	0.097	0.104	0.095	102.7	106	130.3	166
Dataset: zipf1.1 (2.00 × 10^6 elements, 652.2 × 10^3 keys)								
25	0.834	0.201	0.204	0.234	29.2	34	48.8	71
50	0.577	0.127	0.132	0.129	54.4	58	80.4	119
75	0.468	0.116	0.122	0.110	79.6	84	110.9	142
100	0.404	0.107	0.106	0.104	104.5	109	139.8	165
Dataset: zipf1.2 (2.00 × 10^6 elements, 237.3 × 10^3 keys)								
25	0.834	0.209	0.195	0.218	28.5	33	48.0	72
50	0.577	0.147	0.144	0.139	53.7	57	80.5	113
75	0.468	0.120	0.111	0.113	78.8	84	111.4	143
100	0.404	0.098	0.106	0.102	103.9	108	140.3	173
Dataset: zipf1.5 (2.00 × 10^6 elements, 22.3 × 10^3 keys)								
25	0.834	0.210	0.197	0.226	27.2	30	45.2	66
50	0.577	0.141	0.146	0.149	52.1	55	78.9	104
75	0.468	0.124	0.112	0.106	76.9	79	110.5	146
100	0.404	0.100	0.101	0.099	101.9	104	139.1	173

References

E. Cohen. Hyperloglog hyperextended: Sketches for concave sublinear frequency statistics. KDD 2017

R. Kulkarni. A million news headlines [csv data file]. Kaggle, 2017

D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. Journal of the American Statistical Association, 1952

A. Plangprasopchok, K. Lerman, and L. Getoor. Growing a tree in the forest: Constructing folksonomies by integrating structured metadata. KDD 2010

B. Rosén. Asymptotic theory for order sampling. Journal of Statistical Planning and Inference, 1997