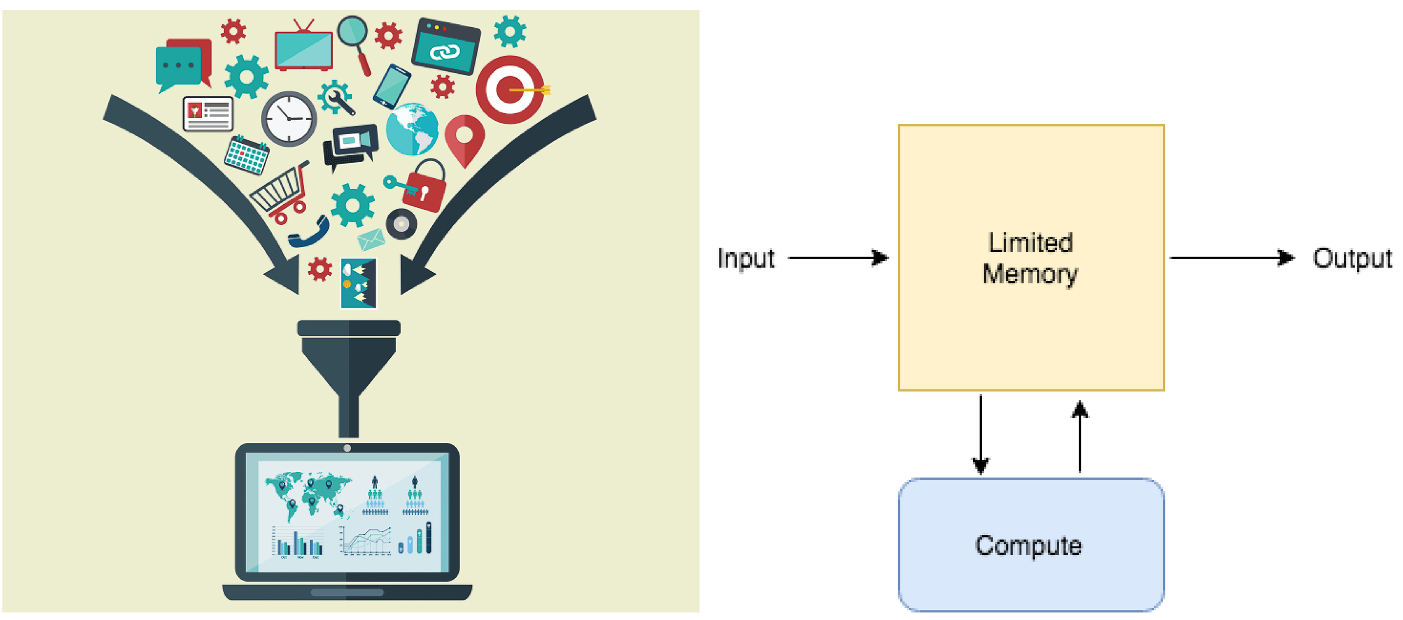
Estimating Entropy of Distributions in Constant Space

Jayadev Acharya[1], Sourbh Bhadane[1], Piotr Indyk[2], Ziteng Sun[1]

[1]Cornell University, [2]Massachusetts Institute of Technology

Motivation

- Small data-hungry computing devices $\rightarrow$ **small storage capability**.



- **Entropy estimation**: Optimal sample complexity given by [1, 2, 3].
- **Streaming algorithms**: Estimate entropy of empirical distribution of a stream [4, 5].
- We initiate study of **memory-sample trade-offs** in statistical inference tasks.

Goal

Given $X_1, X_2, \ldots, X_n \overset{i.i.d}{\sim}$ from a k-ary distribution p and $\varepsilon > 0$, estimate the entropy $H(p)$ upto $\pm\varepsilon$ with probability atleast $2/3$ using a **constant number of words of memory**.

Jargon

Entropy:

$$H(p) := \sum_{x \in \mathcal{X}} p(x) \log(1/p(x)).$$

Sample Complexity: Fewest samples an algorithm requires to solve a statistical inference task.

Word of Memory: $\log k + \log\left(\frac{1}{\varepsilon}\right)$ bits.

Space Complexity: Number of words required to implement an algorithm.

Algorithm	Samples	Space (in words)
Sample-Optimal [1], [2, 3]	$\Theta\left(\frac{k}{\varepsilon \log k} + \frac{\log^2 k}{\varepsilon^2}\right)$	$O\left(\frac{k}{\varepsilon \log k} + \frac{\log^2 k}{\varepsilon^2}\right)$
Streaming [4, 5]	$O\left(\frac{k}{\varepsilon} + \frac{\log^2 k}{\varepsilon^2}\right)$	$O\left(\frac{\log^2(\frac{k}{\varepsilon}) \log\log(\frac{k}{\varepsilon})}{\varepsilon^2}\right)$
General Intervals Algorithm	$O\left(\frac{k \log^2(1/\varepsilon)}{\varepsilon^3}\right)$	20

Table: Sample and Space Complexity for Estimating $H(p)$.

Simple Algorithm

$$H(p) = \sum_{x \in \mathcal{X}} p(x) \log(1/p(x)) = \mathbb{E}_{X \sim p}[\log(1/p(X))].$$

- Draw $X \sim p$.
- Estimate $\log(1/p(X))$ from samples.

Algorithm 1 Simple Algorithm

Require: Accuracy parameter $\varepsilon > 0$, a data stream $X_1, X_2, \ldots \sim p$

Set

$$R \leftarrow O\left(\frac{\log^2(k/\varepsilon)}{\varepsilon^2}\right), \qquad N \leftarrow O\left(\frac{k}{\varepsilon}\right), \qquad S \leftarrow 0.$$

for $t = 1, \ldots, R$ **do**
- $x \leftarrow$ next element in stream.
- $N_x \leftarrow \#$ occurences of x in next N samples.
- $S = S + \log\left(\frac{N}{N_x+1}\right)$.

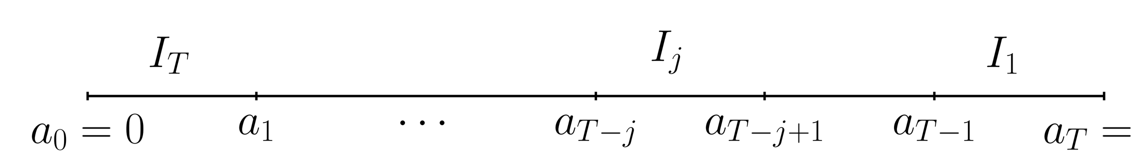
$\hat{H} = S/R$.

Sample Complexity: $(N+1)R = O\left(\frac{k \log^2(k/\varepsilon)}{\varepsilon^3}\right)$ Superlinear :(

Interval Based Algorithms

Simple Algorithm treats each symbol equally. But if $p(x)$ is high, N need not be high.

1. Partition $[0, 1]$ into intervals



2. Randomized algorithm $\mathcal{A}$: $\mathcal{A}(x) = I_j$ w.p. $p_{\mathcal{A}}(I_j \mid x)$.
3. Contributions from each interval:

$$H(p) = \sum_{j=1}^{T} p_{\mathcal{A}}(I_j) H_j, \text{ where } H_j = \mathbb{E}_{X \sim p_{\mathcal{A}(\cdot|I_j)}}[\log(1/p(X))].$$

4. Estimate $p_{\mathcal{A}}(I_j)$, H_j separately using constant words.

Algorithm 2 Interval Based Algorithm

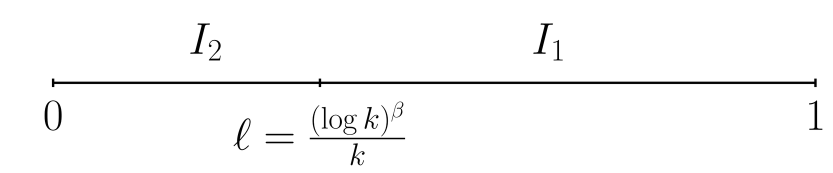
Require: Accuracy parameter $\varepsilon > 0$, a data stream $X_1, X_2, \ldots \sim p$

Set $\{\text{count}_i, S_i = 0\}_1^T$, $\{N_i, R_i,\}_1^T$.

for $i = 1 \cdots T$ **do**
- $\hat{p}_{\mathcal{A}}(I_i) \leftarrow$ Estimate of $p_{\mathcal{A}}(I_i)$.
- **for** $t = 1, \ldots, R_i$ **do**
  - $x \leftarrow$ next element in stream.
  - **if** $\mathcal{A}(x) = I_i$ **then**
    - $\text{count}_i = \text{count}_i + 1$
    - $N_{x,i} \leftarrow \#$ occurences over next N_i samples.
    - $S_i = S_i + \log\frac{N_i}{N_{x,i}+1}$.
- $\hat{H}_i = S_i/\text{count}_i$.

Output $\sum_{i=1}^{T} \hat{p}_{\mathcal{A}}(I_i)\hat{H}_i$.

Two Interval Algorithm



Key Ideas:

- $R_2 < R_1, \quad N_2 > N_1$.
- **Clipping**. Since max probability in I_2 is ℓ, w.h.p $\hat{H}_2$ will be more than $\log\frac{1}{4\ell}$.

$$\hat{H}_2 = \max\left\{\log\left(\frac{N_2}{N_{x,2}+1}\right), \log\frac{1}{4\ell}\right\}.$$

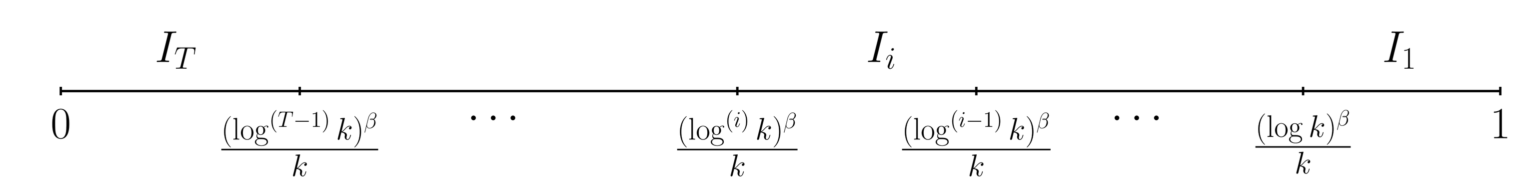
- I_1 : Least probability is $\frac{(\log k)^\beta}{k} \implies N_1 = O\left(\frac{k}{\varepsilon(\log k)^\gamma}\right)$. Range of $\hat{H}_1$ roughly $\log k \implies R_1 = O\left(\frac{\log^2(k/\varepsilon)}{\varepsilon^2}\right)$.
- I_2 : Range of $\hat{H}_2$ roughly $\log(kl) \implies R_2 = O\left(\frac{\log^2(\log k/\varepsilon)}{\varepsilon^2}\right)$. $N_2 = O\left(\frac{k}{\varepsilon}\right)$.

Sample Complexity.

$$O(N_1 R_1 + N_2 R_2) = O\left(\frac{k(\log(\log k/\varepsilon))^2}{\varepsilon^3}\right)$$ Superlinear :(

General Intervals Algorithm

Key Idea. Increase T to $\log^* k = \min_i\left\{i \in \mathbb{N} \text{ s.t. } \log^{(i)} k < 1\right\}$. R_T becomes constant since $1 \le \frac{(\log^{(T-1)} k)^\beta}{k} \le \frac{e^\beta}{k}$.



$$N_i = O\left(\frac{k}{\varepsilon(\log^{(i)} k)^\gamma}\right), \; R_i = O\left(\frac{(\log((\log^{(i-1)} k)/\varepsilon))^3}{\varepsilon^2}\right) \quad i \in [T-1]$$

$$N_T = O\left(\frac{k}{\varepsilon}\right), \; R_T = O\left(\frac{\log^2(1/\varepsilon)}{\varepsilon^2}\right)$$

Future Work

- Lower bounds on space for sample-optimal algorithms ? Is there a sample-optimal algorithm that uses poly$(\log k)$ words of space ?
- Lower bounds on sample complexity of space limited algorithms ?
- Streaming distribution property testing

References

[1] Gregory Valiant and Paul Valiant.
Estimating the unseen: An $n/\log n$-sample estimator for entropy and support size, shown optimal via new CLTs.
In *Proceedings of the 43rd Annual ACM Symposium on the Theory of Computing*, STOC, 2011.

[2] Yihong Wu and Pengkun Yang.
Minimax rates of entropy estimation on large alphabets via best polynomial approximation.
IEEE Trans. Information Theory, 62(6):3702–3720, 2016.

[3] Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman.
Minimax estimation of functionals of discrete distributions.
IEEE Transactions on Information Theory, 61(5):2835–2885, May 2015.

[4] Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor.
A near-optimal algorithm for estimating the entropy of a stream.
volume 6, pages 51:1–51:21, New York, NY, USA, July 2010. ACM.

[5] Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak.
Sketching and streaming entropy via approximation theory.
In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, 2006.