Data Streaming Algorithms for Estimating Entropy of Network Traffic

Ashwin Lall
University of Rochester

Vyas Sekar
Carnegie Mellon University

Mitsunori Ogihara
University of Rochester

Jun (Jim) Xu
Georgia Inst. of Technology

Hui Zhang
Carnegie Mellon University

June 28th, 2006

Outline

- Problem Definition and Applications
- Some Lower Bounds
- The Streaming Algorithm
- The Sieving Algorithm
- Results
- Open Questions, Conclusions, References

The Streaming Model - Notation

- A *stream* is an ordered tuple over the alphabet

$$[n] = \{1, 2, 3, \ldots, n\}.$$

- We denote the number of times that the item i is seen by m_i and the total number of items in the stream by m, i.e., $m = \sum_{i=1}^{n} m_i$.

- The number of distinct items in the stream (i.e., the number of items with non-zero count) is denoted by n_0.

- The tth item in the stream is represented by a_t. So, the stream is represented by $(a_1, a_2, a_2, \ldots, a_m)$, where each $a_t \in [n]$.

The Entropy of a Stream

The *sample entropy* (or just entropy) of a stream is defined to be

$$H \equiv -\sum_{i=1}^{n} (m_i/m) \log (m_i/m).$$

All logarithms are base 2 and, by convention, we define $0 \log 0 \equiv 0$.

The Entropy of a Stream

We will focus on computing the value $S \equiv \sum_{i=1}^{n} m_i \log m_i$ and note that

$$\begin{aligned} H &= -\sum_{i=1}^{n} \frac{m_i}{m} \log\left(\frac{m_i}{m}\right) \\ &= \frac{-1}{m}\left[\sum_i m_i \log m_i - \sum_i m_i \log m\right] \\ &= \log m - \frac{1}{m}\sum_i m_i \log m_i \\ &= \log m - \frac{1}{m} S, \end{aligned}$$

so that we can compute H from S if we know the value of m.

Motivation

- Entropy is a good measure of "diversity"
- Simple volume-based methods are insufficient to detect anomalies
- Several works have suggested entropy as a good metric to detect anomalies (Feinstein et al., Lakhina et al., Wagner et al., Xu et al.)
- Entropy can be used to detect a diverse spectrum of attacks, e.g., denial-of-service, distributed denial-of-service, port scans, worm attacks
- Having access to this information allows us to *profile* the anomalous behavior

Some Lower Bounds

Theorem: Any randomized streaming algorithm to compute the exact value of S when there are at most m items must use $\Omega(m)$ bits of space.

Proof Idea: Reduction from the communication complexity problem of set disjointness.

Theorem: Any deterministic streaming algorithm to approximate S with relative error less than $1/3$ must use $\Omega(m)$ bits of space.

Proof Idea: A counting argument to show that every sublinear algorithm makes an error of at least $1/3$ on some input.

(ϵ, δ)-Approximation

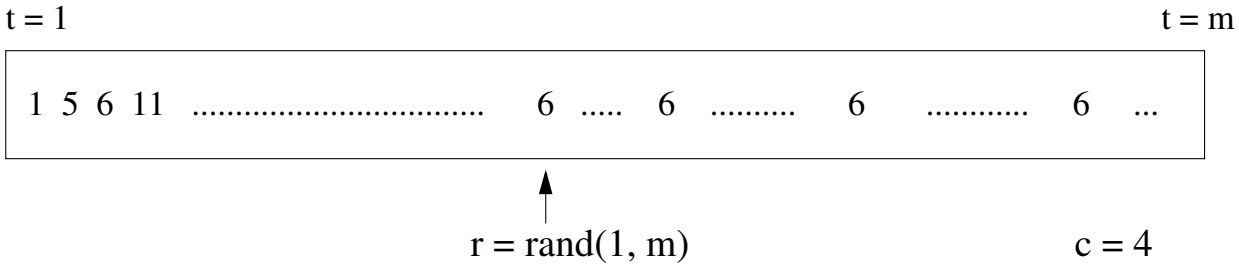
An (ϵ, δ)-approximation algorithm for X is one that returns an estimate X' with relative error more than ϵ with probability at most δ. That is

$$Pr(|X - X'| \geq X\epsilon) \leq \delta.$$

For example, the user may specify $\epsilon = 0.05, \delta = 0.01$ (i.e., at least 99% of the time the estimate is accurate to within 5% error). These parameters affect the space usage of the algorithm, so there is a tradeoff of accuracy versus space.

The Algorithm

The strategy will be to sample as follows:



and compute the following estimating variable:

$$X = m(c \log c - (c-1) \log (c-1)).$$

Algorithm Analysis

This estimator $X = m(c \log c - (c-1) \log (c-1))$ is an unbiased estimator of S since

$$\begin{aligned} E[X] &= \frac{m}{m} \sum_{i=1}^{n} \sum_{j=1}^{m_i} (j \log j - (j-1) \log (j-1)) \\ &= \sum_{i=1}^{n} m_i \log m_i \\ &= S. \end{aligned}$$

Algorithm Analysis, contd.

Next, we bound the variance of X:

$$
\begin{aligned}
Var(X) &= E(X^2) - E(X)^2 \leq E(X^2) \\
&= \frac{m^2}{m}[\sum_{i=1}^{n}\sum_{j=2}^{m_i}(j\log j - (j-1)\log(j-1))^2] \\
&\leq m\sum_{i=1}^{n}\sum_{j=2}^{m_i}(2\log j)^2 \leq 4m\sum_{i=1}^{n} m_i \log^2 m_i \\
&\leq 4m\log m(\sum_i m_i \log m_i) \leq 4(\sum_i m_i \log m_i)\log m(\sum_i m_i \log m_i) \\
&= 4S^2 \log m,
\end{aligned}
$$

assuming that, on average, each item appears in the stream at least twice.

Algorithm contd.

If we compute $s_1 = 32 \log m/\epsilon^2$ such estimators and compute their average Y, then by Chebyschev's inequality we have:

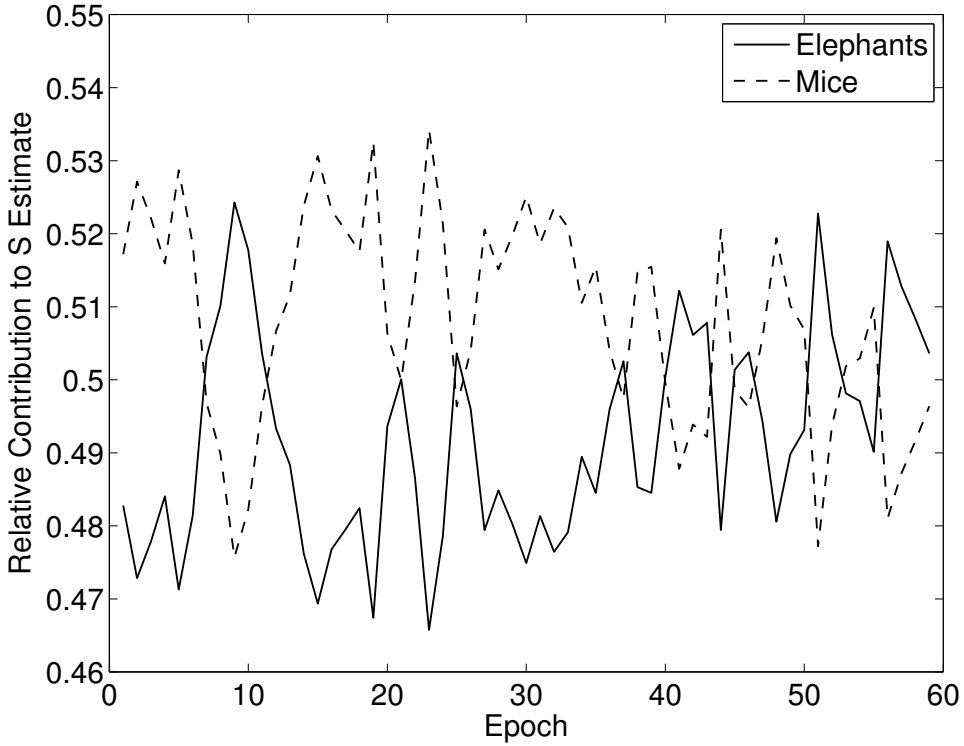
$$\begin{aligned} Pr(|Y - S| > \epsilon S) &\leq \frac{Var(Y)}{\epsilon^2 S^2} \\ &\leq \frac{4 \log m S^2}{s_1 \epsilon^2 S^2} = \frac{4 \log m}{s_1 \epsilon^2} \\ &\leq \frac{1}{8}. \end{aligned}$$

We do this averaging for $s_2 = 2 \log (1/\delta)$ groups. By a Chernoff bound we can get that at least $s_2/2$ averages have relative error at most ϵ with probability at least $1 - \delta$. Hence, the median of averages is an (ϵ, δ)-approximation.

The Sieving Algorithm

- KEY IDEA: Separating out the elephants decreases the variance, and hence the space usage, of the previous algorithm
- Each packet is now sampled with some fixed probability p
- If a particular item is sampled *two or more* times, it is considered an elephant and its exact count is estimated
- For all items that are not elephants we use the previous algorithm
- The entropy is estimated by adding the contribution from the elephants (from their estimated counts) and the ants (using the earlier algorithm)

The Sieving Algorithm, contd.



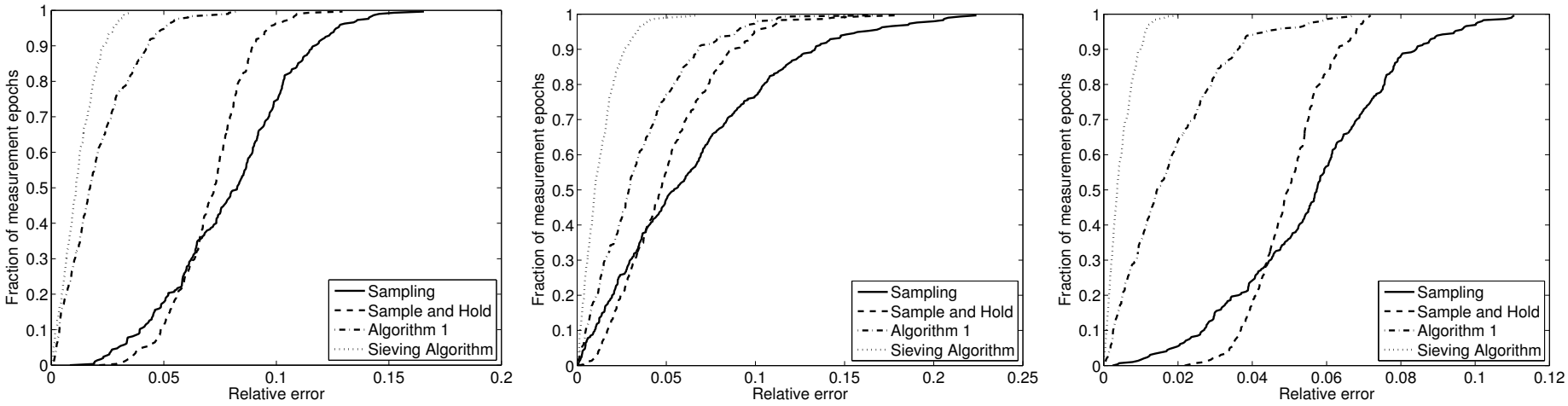
Evaluation Traces

-	Trace 1	Trace 2	Trace 3
Date	Feb. 2, 2004	Aug. 5, 2003	Apr. 24, 2003
Where	USC Los Netto	Mid-sized dept.	UNC Access Link
Epoch Length	1 min.	5 min.	1 min.
Trace Length	60 min.	60 min.	60 min.
# Packets per Epoch	1.7M	0.5M	2.5M
# Distinct Addresses	30,267	2,587	25,565
# Distinct Ports	15,165	4,672	8,080

Algorithms Compared

- Simple sampling
  - Sample packets at rate p
  - Counts are normalized by $1/p$ and used to compute entropy
- Sample and Hold (Estan and Varghese)
  - Sample positions and keep counts from that point on
  - Adjust counts by adding $1/p$ and sum to compute entropy
- Algorithm 1 - The Streaming Algorithm
  - Uses AMS-type estimator
- Algorithm 2 - The Sieving Algorithm
  - Estimates contribution of elephants and ants separately

Evaluation Results



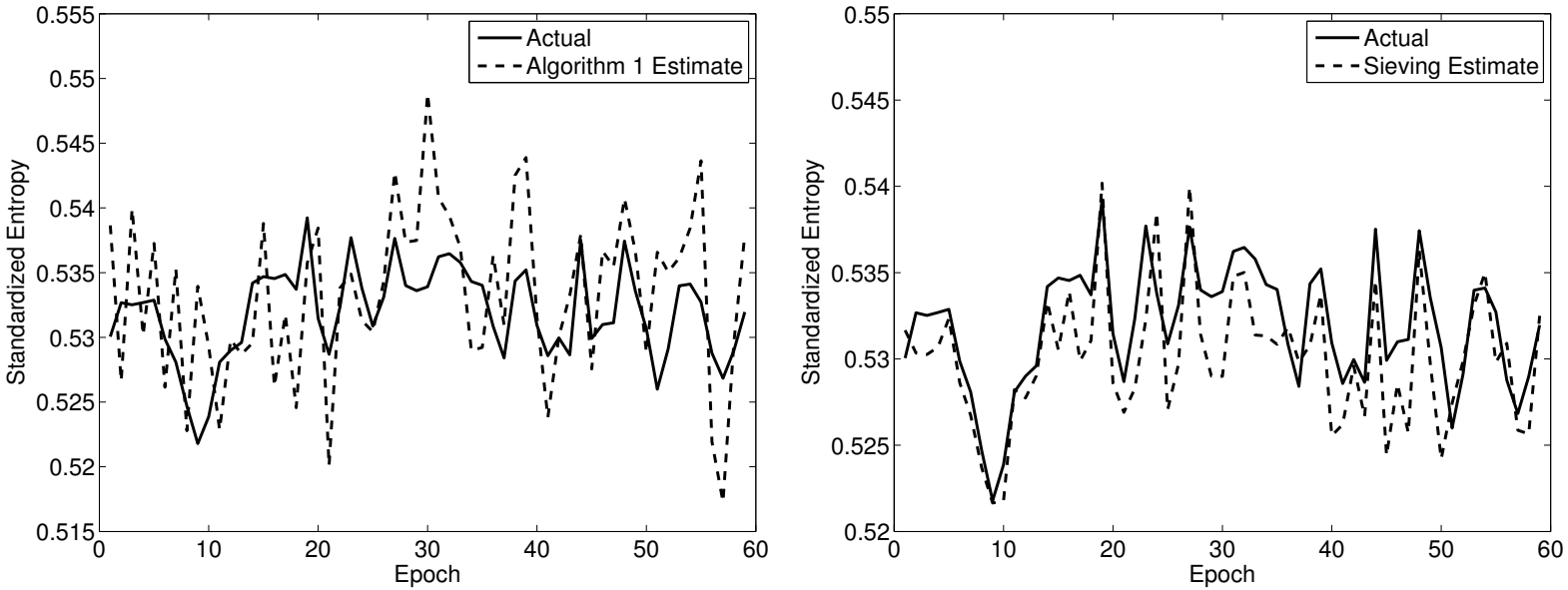
CDF of the three traces on destination address entropy

Evaluation Results

Distribution	Sample	Sample&Hold	Algo. 1	Sieving
DSTADDR	0.198	0.192	0.013	0.013
SRCADDR	0.054	0.049	0.017	0.004
DSTPORT	0.116	0.109	0.016	0.015
SRCPORT	0.069	0.062	0.016	0.005

Trace 3: Mean relative error in S estimate

Evaluation Results



Verifying the accuracy in estimating the standardized entropy

Open Problems and Conclusions

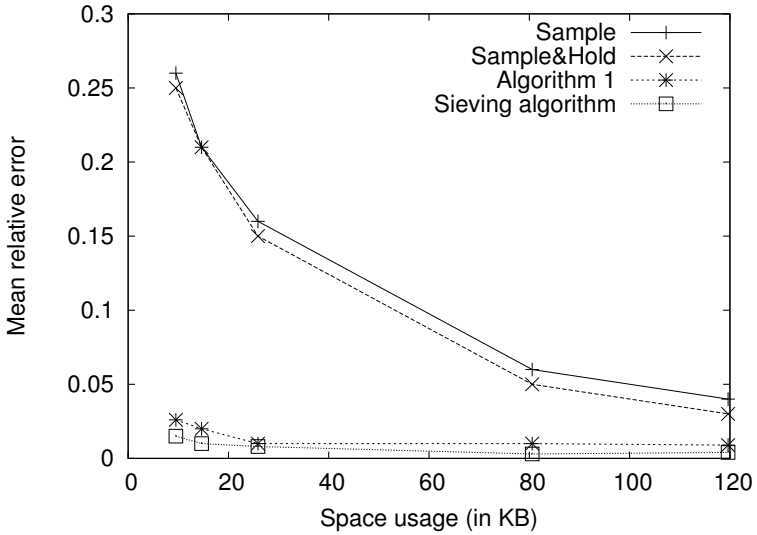
- There is much to be learnt about *streaming algorithms* and how they outperform traditional sampling methods
- The idea of separating out the elephants and performing computations for them and the rest of the stream separately may have a much wider use
- Are there methods out there to improve our results further? Can we do better if we make slightly stronger assumptions about the distribution of the stream?
- How hard would it be to maintain the entropy of arbitrary subpopulations? For example, to allow a network operator to study the behavior of a single source address *after* the data has been collected.

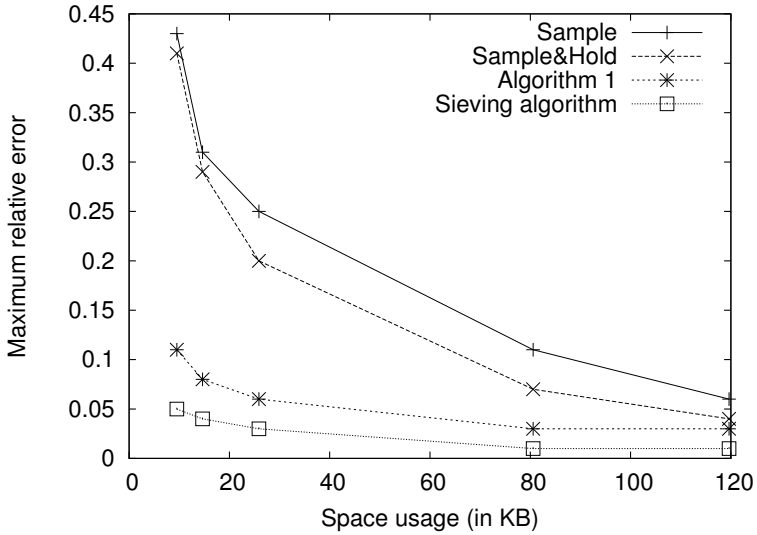
References

- N. Alon, Y. Matias, M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the 28th annual ACM Symposium on Theory of Computation (STOC 1996)*, New York, NY, USA, 1996. ACM Press.
- A. Chakrabarti, K. Do Ba, S. Muthukrishnan. Estimating entropy and entropy norm on data streams. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS 2006)*, Marseille, France, 2006. Springer LNCS.
- B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. In *SIAM Journal of Discrete Mathematics*, 5(4):545-557, 1992.

Thanks for your attention!

Extra slides - Varying Sampling Rate





Extra slides - Threshold for Sieving