Streamed Approximate Counting of Distinct Elements

Beating Optimal Batch Methods

Daniel Ting
Facebook
1730 Minor Avenue
Seattle, WA
dting@fb.com

ABSTRACT

Counting the number of distinct elements in a large dataset is a common task in web applications and databases. This problem is difficult in limited memory settings where storing a large hash table table is intractable. This paper advances the state of the art in probabilistic methods for estimating the number of distinct elements in a streaming setting New streaming algorithms are given that provably beat the "optimal" errors for Min-count and HyperLogLog while using the same sketch.

This paper also contributes to the understanding and theory of probabilistic cardinality estimation introducing the concept of an area cutting process and the martingale estimator. These ideas lead to theoretical analyses of both old and new sketches and estimators and show the new estimators are optimal for several streaming settings while also providing accurate error bounds that match those obtained via simulation. Furthermore, the area cutting process provides a geometric intuition behind all methods for counting distinct elements which are not affected by duplicates. This intuition leads to a new sketch, Discrete Max-count, and the analysis of a class of sketches, self-similar area cutting decompositions that have attractive properties and unbiased estimators for both streaming and non-streaming settings.

Together, these contributions lead to multi-faceted advances in sketch construction, cardinality and error estimation, the theory, and intuition for the problem of approximate counting of distinct elements for both the streaming and non-streaming cases.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Probabilistic algorithms, Stochastic processes; H.2.8 [**Database Applications**]: Data mining

Keywords

distinct elements, martingale, cardinality estimation, randomized algorithms

KDD'14, August 24–27, 2014, New York, New York, USA.
ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623669.

1. INTRODUCTION

Approximating the number of distinct elements in a dataset while using a small amount of memory is an important problem with broad industrial applications. A web company may wish to count the number of distinct users accessing a service broken down by country of origin [12]. Other areas of application include counting network flows to detect denial of service attacks [6] and database warehousing [1].

A number of algorithms exist for approximately estimating these counts. Most tackle the problem by first constructing a summarization of the data or *sketch* and then estimating the number of distinct elements, the cardinality, from the sketch. In practice, Flajolet and Martin's HyperLogLog [8] and Chen et al's Self-learning bitmap [4] (S-bitmap) constitute the state of the art in the field when there is no prior estimate of the cardinality. Several theoretical results exist as well. Kane et al [11] give an algorithm with optimal space complexity but sub-optimal statistical efficiency while Giroire's Min-count algorithm [9] has optimal statistical efficiency but poor space requirements. Although these algorithms have been described as streaming algorithms, only the S-bitmap exploits the streaming property of data to achieve better accuracy.

This paper develops a general recipe for constructing streaming cardinality estimation algorithms by introducing the area cutting process and an estimator for the cardinality using martingale methods. The martingale methods extend the estimation methods used by the S-bitmap [4]. This recipe covers almost all sketching methods. Any sketch which is generated by hashing of each element in the data stream and is not affected by duplicates elements has a representation as an area cutting process. The associated martingale estimator is provably unbiased and provably optimal if queried immediately after a sketch update.

Streaming versions of HyperLogLog and Min-count are shown to have superior performance to their non-streaming counterparts. Regular HyperLogLog is shown to require approximately 1.5 times more bits than streaming HyperLogLog to achieve the same error, and Min-count requires twice the size of streaming Min-count. These streaming estimators are able to beat optimal estimators which use only the final sketch is because the streaming estimators are able to exploit the values of intermediate sketches as well. Furthermore, unlike the S-bitmap which can only be used in a streaming setting, these sketching methods have the advantage that they may also be used in distributed settings where results may be need to be aggregated over multiple servers or computed within a map-reduce framework.

A special class of decomposable self-similar area cutting processes is introduced and shown to have attractive properties. They are shown to have an unbiased estimator for non-streaming data as well as analytic formulas for the variance for both the streaming and non-streaming case. A new sketch construction, Discrete Max-count, yields a decomposable self-similar area cutting process which has good performance for small sketches. For a non-optimal encoding of the Discrete Max-count sketch, the space-complexity is derived and shown to have no term which jointly depends on both the error ϵ and cardinality n.

The empirical performance of each method is evaluated as well using both simulations and an anonymized Facebook test dataset. Unlike HyperLogLog which requires special corrections for small and medium sized cardinalities [10], the performance of the estimators is consistently good throughout the entire range of cardinalities tested.

Several ideas of note are introduced in this paper. To introduce these ideas the paper is structured as follows. In section 2, we formulate the cardinality estimation problem by describing the probabilistic data generating process for the sketches. This leads to the martingale estimator and its properties, including its optimality properties. Next, several existing sketches are described and shown to yield streaming estimates. The area cutting process and a geometric interpretation of the sketches are give in section 5. This paper then focuses on self-similar area cutting processes and their properties. Discrete Max-count is introduced as a special case and its space complexity is analyzed, and the optimal and near optimal estimators for the non-streaming versions of Min-count and HyperLogLog are shown to be non-optimal in streaming settings. The streaming methods are evaluated via simulation in section 7 and future research directions are discussed.

2. PROBABILITISTIC COUNTING

To fix the notation in this paper, let $\mathcal{X} = (X_1, X_2, ...)$ be a stream of elements and n denote the cardinality, the number of distinct elements, of $\mathcal{X}$. Cardinality estimation methods involve constructing a sequence of sketches S_i which are functions of the data seen up to some point, and then estimating the cardinality $\hat{N}(S_i)$ from the sketches. To ensure that only distinct elements are counted, the sketches should not be affected by duplicates. In other words, they should have the property $S_i = S_{i-1}$ if $X_i = X_j$ for some $j < i$. In most probabilistic methods, the sketches are based on a transformed stream of data which has been converted using a strong universal hash function h so that $(h(X_{i_1}), h(X_{i_2}), ...)$ forms a sequence of independent $Uniform(0, 1)$ random variables where the subsequence (i_j) picks out the first occurrence of each distinct element. The hash function ensures that duplicates remain duplicates after transformation. The sketches S_i are then random elements from some distribution parameterized by n_i, the number of distinct elements encountered up to the i^{th} element in the stream. Thus, the problem of estimating the cardinality is reduced to the statistical problem of estimating a parameter for a known family of distributions given the data S_i.

Our insight for exploiting the streaming nature of the data is that using just the last sketch S_{last} to estimate the cardinality throws away information contained in the entire sequence of sketches. We augment the sketch by including the current estimate into the sketch $\tilde{S}_i = (S_i, \hat{N}_i)$. This augmentation allows the construction of a provably unbiased martingale estimator with just an additional $O_p(\lg n)$ bits. Here, the notation $O_p(\lg n)$ is the probabilistic analog of $O(\lg n)$ where the bound holds for arbitrarily small probabilities.

Algorithm 1 Algorithm for updating an existing estimate $\hat{n}_{prev}$ and sketch with a new item.

```
function UpdateEstimate(n̂_prev, sketch, item_new)
    q ← P(sketch is modified)
    Add item_new to sketch
    if sketch is modified then
        return n̂_prev + 1/q
    else
        return n̂_prev
    end if
end function
```

2.1 Counting with Markov processes and Martingales

The key idea behind all streaming algorithms described in this paper is that a Markov chain is formed by the sequences of sketches generated from a data stream. This Markov chain is obtained by taking the original sequence of sketches $S_i^{(orig)}$ indexed by the number of items i encountered and considering only the subsequence S_t indexed by the number of *distinct* items t encountered.

THEOREM 1. *The sequence of sketches (S_t) forms a Markov chain when h is a strong universal hash.*

PROOF. The next sketch $S_{t+1} = f(S_t, h(X_t))$ is some function of the current sketch and the next hashed value. Since h is a strong universal hash, the $h(X_t)$ are independent and identically distributed, and the probability of transitioning to some value depends only on the previous value of the sketch S_t □

For ease of exposition, this paper will refer to the index t interchangeably as both time and the number of distinct elements encountered.

Define the *martingale estimator* to be the Markov chain

$$\hat{N}_{t+1} = \hat{N}_t + \frac{1}{q(S_t)} 1(S_{t+1} \neq S_t) \tag{1}$$

where $q(S_t)$ is the probability of modifying the current sketch and transitioning out of the current state. The initial state of the chain is $\hat{N}_0 = 0$. The idea behind this estimator is that with one time step the expected increase in the estimator is exactly 1. Note that although the underlying sequence (S_t) is not observed since the number of distinct items t is not known at an arbitrary location in the stream, the estimator depends only on observable events, namely on modifications to the sketch.

The resulting algorithm is simple and elegant and can be applied to any sketch which is unaffected by duplicates. Given a sketch, augment it with a counter c and add $1/q(S_{T_{t-1}})$ every time there is a change in the sketch. Note that the estimate is never updated when a duplicate item is encountered since duplicates do not modify the sketch. The algorithm is summarized in algorithm 1.

The name of the estimator comes from the fact that $\hat{N}_t - t$ is a martingale. A martingale M_t is a Markov process such that $\mathbb{E}(M_{t+1}|M_t) = M_t$. The martingale property aids in

Symbol	Description
n	True cardinality
m	Number of bins or samples
S_t	Sketch after t distinct items are encountered
T_i	Time of the i^{th} modification of the sketch
$q(S_t)$	Probability that a new item modifies the sketch S_t
q_i	$q(S_{T_i})$
(Δ_t, Y_t)	offset and samples of an offset decomposition

Table 1: Table of symbols

deriving properties of the estimator. The most important property is given by the following theorem.

THEOREM 2. *The process $M_t = \hat{N}_t - t$ is a martingale. Hence, $\hat{N}_n$ is an unbiased estimate of the cardinality n.*

PROOF. The martingale property is trivial to check. $\mathbb{E}M_t = \mathbb{E}\mathbb{E}(M_t|M_{t-1}) = \mathbb{E}M_{t-1}$. By induction, $\mathbb{E}M_t = \mathbb{E}M_0 = 0$. In other words, $\mathbb{E}\hat{N}_n = n$, and $\hat{N}_n$ is unbiased. □

Another way to view the estimator comes from decomposing it in terms of holding times for the Markov chain. Let $T_1, T_2, \ldots$ be the jump times of the chain, so that T_i be the time of the i^{th} modification to the sketch. The holding time $T_i - T_{i-1}$ is the time the chain remains in state $S_{T_{i-1}}$. Let T_K be the sketch's last modification time before time n.

$$n = \sum_{T_i \le n} (T_i - T_{i-1}) + (n - T_K) \tag{2}$$

$$\hat{N}_t = \sum_{T_i \le n} \mathbb{E}(T_i - T_{i-1}|S_{T_{i-1}}) = \sum_{T_i \le t} \frac{1}{q(S_{T_{i-1}})}. \tag{3}$$

Since the holding time $T_i - T_{i-1}$ of a discrete Markov chain has a $Geometric(q(S_{T_{i-1}}))$ distribution, the estimator is incremented by precisely the expected time it takes before the sketch is modified.

For convenience, we provide a table of commonly used symbols in table 1.

2.1.1 Bayesian approach

We also consider a Bayesian approach to estimation. This leads to subtlely different estimators which are biased but have slightly stronger optimality guarantees when the cardinality is drawn from a $Geometric(\alpha)$ prior. The Bayesian formulation is also of additional interest when estimating the error in section 2.2 since the posterior is tractable. The Bayesian estimate with lowest squared error given the sequence of observed sketches is the conditional expectation given the sketches

$$\hat{n}_{Bayes} = \mathbb{E}(N|S_{T_1}, \ldots, S_{T_K}, K) \tag{4}$$

$$= \sum_{t=1}^{K} \mathbb{E}(T_t - T_{t-1}|S_{T_{t-1}}) + \mathbb{E}(N - T_K|S_{T_K}, K). \tag{5}$$

Interestingly, this gives a slightly different decomposition than the one for the martingale estimator described by 3. The additional term $\mathbb{E}(N - T_K|S_{T_K}, K)$ means that the estimator is biased even under a uniform prior on N.

Under a $Geometric(\alpha)$ prior on the cardinality N, it is easy to derive

$$T_t - T_{t-1}|S_{T_{t-1}} \sim Geometric(1 - (1-\alpha)(1-q_{t-1})) \tag{6}$$

$$N - T_K|S_{T_K}, K \sim Geometric(1 - (1-\alpha)(1-q_{T_K})) \tag{7}$$

where $q_t = q(S_{T_t})$ and the holding times are conditionally independent given the sketches. Note that as $\alpha \to 0$ the prior approaches the improper uniform prior and the conditional distributions simplify to $Geometric(q(S_T))$ for appropriate T. In this case, the estimator is identical to the martingale estimator except that the Bayesian estimator has one additional term for the last holding time. The Bayesian conditional mean estimator is given by

$$\hat{N}_{Bayes} = \sum_{i=0}^{K} \frac{1}{q_i + \alpha(1 - q_i)}. \tag{8}$$

2.2 Error estimates

It is possible to maintain an estimate of the error in addition to keeping an estimate of the cardinality. We estimate the error using both a martingale approach and a Bayesian approach and describe the properties of each.

The martingale estimate of the variance gives an unbiased estimate of the variance, but only at jump times when the sketch is modified. The quadratic variation $\langle M_t \rangle$ of a process (M_t) is defined to the the sum of its squared increments:

$$\langle M_t \rangle = \sum_{i=0}^{t-1} (M_{i+1} - M_i)^2. \tag{9}$$

It follows from Doob's decomposition [5] that if M_t is a martingale then the quadratic variation process $\langle M_t \rangle$ is the unique predictable process such that $M_t^2 - \langle M_t \rangle$ is a martingale. In other words, since $\mathbb{E}M_t^2 = \mathrm{Var}(M_t)$, the quadratic variation process is an unbiased streaming estimate of the variance.

For the martingale estimator, the quadratic variation conditional on the observed sketches yields an unbiased estimate of the martingale estimator's variance at jump times:

$$\widehat{\mathrm{Var}}(\hat{N}_{T_k}) = \sum_{i=1}^{k} \mathrm{Var}(T_i - T_{i-1}|q_{i-1}) = \sum_{i=1}^{k} \frac{1 - q_i}{q_i^2} \tag{10}$$

where the second line uses the fact that the increments are geometrically distributed and that the variance of a $Geometric(q)$ random variable is $\frac{1-q}{q^2}$,

The reason this estimator is inadequate for estimating the variance at a fixed cardinality n, is that the increments after the last jump time involve the unobservable quantity $\sum_{t=T_K+1}^{n} 1^2 = (n - T_K)$. We may get around this by applying a Bayesian argument. Given a uniform prior on $n - T_k$, the posterior is given by $n - T_K \sim Geometric(q_K)$. This gives us a slightly modified estimate

$$\widehat{\mathrm{Var}}(\hat{N}) = \sum_{i=1}^{k-1} \frac{1 - q_i}{q_i^2} + \frac{1}{q_k}. \tag{11}$$

A fully Bayesian approach is simple to analyze. Under a Bayesian model with a $Geometric(\alpha)$ prior, all of the holding times, including the last difference, are independent and geometrically distributed according to equations 6 and 7. The variance estimate for the Bayesian estimator at a fixed cardinality is

$$\widehat{\mathrm{Var}}(\hat{N}_{Bayes}) = \sum_{i=0}^{K} \frac{1 - (q_i + \alpha(1 - q_i)}{(q_i + \alpha(1 - q_i))^2} \tag{12}$$

Note that if the estimate is at a stopping time, the Bayesian estimate for the variance also drops the last term in the

sum. If $\alpha = 0$ and the estimate is at a jump time, the Bayesian estimate exactly matches the martingale estimator. However, if the estimate is at a fixed time n, the contribution of the last difference $N - T_K$ is $\frac{1-q_K}{q_K^2}$, the variance of a $Geometric(q_K)$ random variable, rather than $1/q_K$, the mean of a $Geometric(q_K)$ random variable and the contribution of the last term for the martingale estimator.

3. OPTIMALITY

3.1 Statistical efficiency versus space-complexity

The optimality results in this paper primarily address the notion of statistical efficiency. Given a sketch construction which yields a data generating distribution, what is the estimator with the lowest variance or risk? Much of the computer science theory literature focuses on optimal space-complexity. The primary differences between these two notions of optimality are three-fold. First, space-complexity is concerned with an asymptotic rate while statistical efficiency is concerned with the exact constant governing the rate. Second, statistical efficiency is always analyzed with respect to a given data generating distribution or the sketch construction in this case, while optimal space-complexity does not assume a particular construction. Third, space-complexity depends on both the estimation method from data and the encoding of the data while statistical efficiency only deals with estimation.

Although the method given by [11] has optimal space complexity, it has non-optimal statistical efficiency and a non-trivial probability of not returning any result whatsoever. Empirical results [13] show that HyperLogLog has lower error for the same sketch size. Because of these reasons, we do not consider finding an optimal algorithm for cardinality estimation to be a closed problem.

Given the current state-of-the-art in cardinality estimation, statistical efficiency has several advantages as a notion of optimality for practical algorithms. Since any decent statistical estimator for a parametric distribution has standard deviation on the order of $O(1/\sqrt{m})$ where m is the number of observations, the asymptotic rate of the standard deviation is typically meaningless. Statistical efficiency appropriately addresses the estimation problem while leaving the efficient encoding of the sketch as a separate issue. Furthermore, methods like HyperLogLog achieve a space-complexity of $O(\epsilon^{-2} \log\log n + \log n)$ compared to the optimal space-complexity $O(\epsilon^{-2} + \log n)$. The $\log\log n$ term is effectively constant as 6 bits allows estimation of cardinalities approaching 2^{64}. If the $\log\log n$ is ignored, the space-complexity matches the optimal one.

3.2 Optimality of the estimators

We establish that both the martingale estimator and its error estimate are optimal in the sense that they are minimum variance unbiased estimator at jump times. In other words, no other algorithm can also use the same sketch construction and have lower squared error if it gives an unbiased estimate. In Bayesian settings, the Bayesian cardinality estimators are optimal for both fixed cardinalities and at jump times when the cardinality is drawn from the prior distribution.

First, we consider the optimality of the martingale estimators at jump times for an infinite stream of data. Conditional on the sequence of sketches (S_i), the increments $T_{t+1} - T_t|(S_i)$ are independent random variables. This implies that the estimator $\hat{n}_T$ at a jump time T is the true conditional mean of the jump time given the sequence of sketches, and the martingale estimator of the variance is the true conditional variance. Hence, they are the optimal unbiased estimate under mean squared error. By the same reasoning, the variance estimate for the martingale estimator is also optimal at jump times. In the fixed n case, we are not able to prove that the estimator is a minimum variance unbiased estimator for fixed n.

In the Bayesian case, when the cardinality is a draw from the prior distribution, the Bayesian posterior mean is always optimal since it is the minimizer of squared error. However, it is biased, and an appropriate prior distribution is generally not known. For the same reason, the variance estimate for the Bayesian estimator is also optimal under these conditions.

4. EXISTING METHODS AND THEIR STREAMING COUNTERPARTS

The previous section presented universal estimators of the cardinality and error for any Markov chain of sketches. In this section, these estimators are applied to existing sketch constructions. HyperLogLog [8] and Min-count [9] are described and their streaming counterparts are derived using the martingale estimator. We also describe the S-bitmap streaming algorithm which already uses a martingale estimator and show that the martingale estimator for Linear Probabilistic Counting is nearly identical to the original non-streaming version. Furthermore, we show that the martingale estimator is optimal in non-streaming settings as well.

4.1 HyperLogLog

The HyperLogLog algorithm [8] estimates the cardinality based on the Flajolet-Martin (FM) sketch. The sketch is an integer valued m-vector $S_i = (S_{i1}, ..., S_{im})$. The sketch is highly compact since each vector entry may be represented in $O_p(\lg\lg(n/m))$ bits. For the FM-sketch, each element X_i is translated into a random pair

$$\begin{aligned} X_i &\to f(U_i) = (M_i, Y_i) \\ M_i &\sim Uniform([m]) \\ Y_i &\sim Geometric(1/2). \end{aligned}$$

The sketch updates bin M_i by taking the max of the current value in the bin and the new value Y_i

$$S_{i,j} = \begin{cases} \max\{S_{i-1,j}, Y_i\} & \text{if } j = M_i \\ S_{i-1,j} & \text{else} \end{cases}.$$

Sketch S_i is updated in one time step with probability

$$q(S_i) = \frac{1}{m}\sum_{j=1}^{m} P(Y_i > S_{ij}) = \frac{1}{m}\sum_{j=1}^{m} 2^{-S_{ij}}. \qquad (13)$$

The HyperLogLog estimator is $\hat{N}_{hyperloglog} = \alpha/q(S_n)$ where S_n is the final sketch and α is an appropriate constant to correct for bias. Section 6.4 show that this modification yields a reduction in the variance by a factor of ≈ 1.56.

4.2 Min-count

Giroire's Min-count algorithm [9] maintains the m^{th}-order statistics of the observed sample. In other words, it keeps the m smallest elements in the sample. Let $S^{(m)}$ denote the

m^{th} smallest element in S. Without loss of generality, assume the hashed values are distributed $Uniform(0,1)$. The probability a sketch S_i is updated by a new element is

$$q(S_i) = S_i^{(m)}. \tag{14}$$

Like HyperLogLog, the non-streaming Min-count estimator of the cardinality is only a function of the final update probability $q(S_n)$ of the final sketch. The estimator is given by $\hat{N}_n^{(mincount)} = \frac{m-1}{q(S_n)}$. This estimator is provably optimal [3] when only the final sketch is observed. However, section 6.2 shows that the streaming estimator provably beats this optimal estimator.

4.3 Linear probabilistic counting

The Linear Probabilistic Counting algorithm (LPCA) [15] uses a simple bitmap as a sketch. Each item in the data stream is hashed to a bit, and that bit is set if it is not already. This algorithm is poor for large counts, as it requires space roughly linear in n. If B_t is the total number of bits set at time t, then the probability of transitioning out of the current state is $q(S_t) = 1 - B_t/m$. In this case, the martingale estimator has a simple analytic expression. This may be compared to the usual LPCA estimator.

$$\hat{N}_n^{(Harmonic)} = \sum_{i=1}^{B_n} \frac{m}{m-i} = m(H_m - H_{m-B_n}) \tag{15}$$

$$\hat{N}_n^{(LPCA)} = -m \log\left(1 - \frac{B_n}{m}\right) \tag{16}$$

where H_i denotes the i^{th} harmonic number. Since $\log i \approx H_i$, the LPCA estimator is a very good approximation to the martingale estimator.

The martingale estimator is, in fact, provably optimal for both streaming and non-streaming cases. Since the estimator is a function of the number of bits set, B_n, in the final sketch and the sketch is not affected by the order in which elements are encountered, it is applicable in non-streaming settings. Furthermore, B_n uniquely determines the likelihood function, and hence, is a minimal sufficient statistic. Since the martingale estimator is an unbiased estimator and a function of a minimal sufficient statistic, the Rao-Blackwell theorem states that it is a minimum variance unbiased estimator [2].

4.4 S-bitmap

The S-bitmap algorithm also uses a simple bitmap as a sketch. The special property of the S-bitmap is that the relative error of the cardinality estimate defined to be $\sqrt{\mathrm{Var}(\hat{N})/N}$ is constant over a chosen range. Like LPCA, the S-bitmap is highly efficient for smaller cardinalities. Table 2 in [4] shows that it often requires less space to achieve a desired error for cardinalities up to 10^7. One downside of the S-bitmap is that it is not applicable to distributed settings because the value of sketch S_i depends on the order in which the distinct elements are observed.

The derivation of the S-bitmap directly chooses transition probabilities $q(S_i)$ that yield a constant relative root mean squared error.

$$X_i \to (M_i, U_i)$$
$$M_i \sim Uniform([m])$$
$$U_i \sim Uniform(0,1)$$

The sketch is updated by setting bit M_i if that bit is empty and $U_i > c(B_{i-1})$ for a carefully chosen cutoff function c that is monotonically increasing. The cutoff function is chosen in a manner that ensures that the relative error is constant up to the capacity of the sketch. This update process is further illustrated in section 5 and figure 2.

The S-bitmap estimator is identical to the martingale estimator. Like LPCA, the estimator is purely a function of the number of positive bits B_n in the final sketch, and B_n is a minimal sufficient statistic. Thus, the S-bitmap estimator is an optimal streaming estimate.

5. AREA CUTTING PROCESSES: A GEOMETRIC INTERPRETATION

So far, this paper has shown how the theory of Markov chains and martingales lead to cardinality estimators and their properties. We now give a geometric interpretation of sketch constructions via the *area cutting process*. This process is universally applicable as it can describe any sketch for counting distinct elements which is not affected by duplicates. It is useful as it can be used to create new sketches as well as to analyze existing ones.

Consider a Markov process that cuts out sections of a unit square. At each time step, a point is uniformly drawn from the unit square. Rather than hashing to a single random variable, hash each element to a pair of independent uniforms, $X_i \to (U_i, U_i')$. If that point falls on a portion of the square that has already been cut off, then do nothing. Otherwise, apply a deterministic procedure to cut off some section of the square that contains that point. It is easy to see that a duplicated point cannot induce more than one cut, so the sketch is not affected by duplicates and approximate for counting distinct items. The probability of an update at step i is equal to the remaining uncut area A_i.

THEOREM 3. *All sketches that are not affected by duplicates have a representation as an area cutting process whenever the universe from which elements are drawn is measureable.*

PROOF. The uncut area is simply the measure of the remaining elements which can affect the sketch. □

THEOREM 4. *The area process forms a sufficient statistic for the unknown cardinality n.*

PROOF. Write down the log-likelihood of the process. The only terms that depend on the holding times $T_i - T_j$ are functions of the area. □

Together with the likelihood and sufficiency principles [14], theorems 3 and 4 say that for any sketch construction, all information that is useful for estimating the cardinality is contained in the area process. More specifically, the Rao-Blackwell theorem [2] states that if $\hat{n}$ is an unbiased estimator and the sequence of areas (A_{T_i}) is a sufficient statistic, then $\mathbb{E}(\hat{n}|A_{T_1}, A_{T_2}, \ldots, A_n)$ has variance no worse than $\hat{n}$. Formally, this means that any estimator which is not a function of the area process can be improved by turning it into a function of the area process by taking the conditional mean.

5.1 Existing methods as area cutting processes

To illustrate the area cutting process, we describe several existing sketch constructions as area cutting processes.

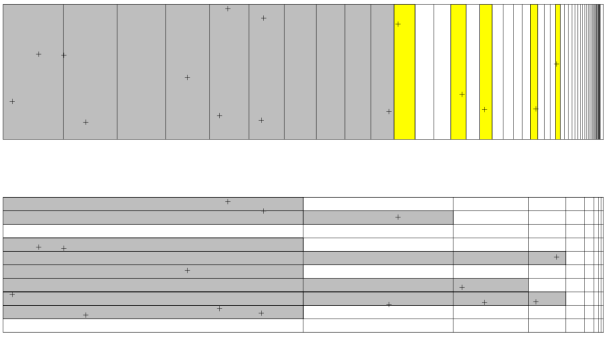

Figure 1: Pictoral representation of the Discretized Max-count (top) and FM (bottom) sketch. The + signs represent distinct items that are hashed uniformly to the rectangle. As each point is placed on the rectangle, shade an area covering the point if it is not already shaded. The shaded area represent the areas that are cut out of the sketch. New points that land in a shaded region do not affect the sketch. For the Discrete Max-count sketch, everything to the left of the cutoff is shaded regardless of whether or not a point falls in a particular box.

5.1.1 Hyperloglog

First, consider a continuous process corresponding to the FM-sketch used in HyperLogLog. Divide the unit square into m horizontal strips of equal size. The point (U_i, U_i') picks out a strip which contains the point and then cuts off everything to the left of U_i within that strip. At any given time, the remaining area in strip j is equal to the one minus the maximum of the U_i that fall into that strip.

A discretization of this continuous process leads to the FM-sketch. For each horizontal strip, divide it along the vertical axis into intervals $[2^v, 2^{v-1})$ indexed by the exponent v. Rather than store the exact location Z_j of the maximum element in the strip, store the exponent of the interval it belongs to. Equivalently, store $\lfloor -\lg Z_j \rfloor$. Since $\lfloor -\lg U_i \rfloor \sim Geometric(1/2)$ when $U_i \sim Uniform(0,1)$, this is equivalent to taking the maximum of $Geometric(1/2)$ random variables. Figure 1 illustrates this sketch as an area cutting process.

5.1.2 Min-count

The Min-count sketch is even simpler to visualize as an area cutting process. Instead of a unit square, the process cuts a unit interval. Everything greater than the m^{th} sample is cut off. Note that each of the m samples that are stored also cut off the exact location of the unit interval on which it falls. That prevents duplicates from affecting the sketch. However, those locations form a set of measure 0 since two independent $Uniform(0,1)$ variables are equal with probability 0. This also highlights the primary weakness of the Min-count algorithm. In order to have enough precision to treat each sample as occupying an area of 0 measure, each sample requires many bits to store.

5.1.3 Linear probabilistic counting

Divide the unit square into m uniform vertical strips. Let each item in the stream hash to a uniformly distributed point on the square. Cutting out the vertical strip that an item falls into is equivalent to setting that bit. Figure 2 illustrates the difference between the S-bitmap and LPCA sketch.

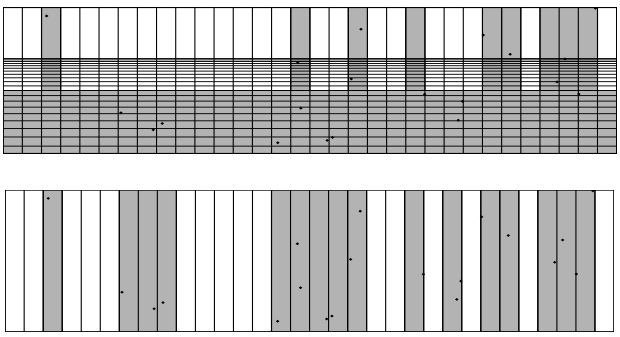

Figure 2: Pictoral representation of the (top) S-bitmap, (bottom) LPCA. Both share the same set of items that are hashed uniformly on the rectangle, but each cuts out a different area. Each vertical cut or equivalently, each bit that is set, results in a horizontal cut as well. Due to this, the S-bitmap sketch has fewer bits set and more remaining capacity.

5.1.4 S-bitmap

The area cutting process for the S-bitmap sketch is similar to the LPCA sketch. In addition to cutting a vertical strip out, every update to the square also cuts out an additional horizontal strip. Figure 2 illustrates this process. By cutting out a larger area with each bit, the capacity of the sketch is increased by extending the time until the next bit is set.

5.1.5 Virtual and Multiresolution bitmap

The virtual bitmap counting algorithm is the same as LPCA except that with probability $1-\alpha$ an element is hashed to a null bucket and does not update the sketch. This is equivalent to starting with $1-\alpha$ of the unit square already cut out and dividing the remaining area into m vertical strips like linear probabilistic counting. This allows the sketch to work for larger cardinalities at the cost of worse estimation for smaller cardinalities. The unbiased martingale estimator is simply $1/\alpha$ times the martingale LPCA estimator.

The multiresolution bitmap [6], covers a range of cardinalities effectively by using multiple virtual bitmaps. Similarly to all the other bitmap based sketches, it cuts out vertical strips corresponding to the bin each element hashes to. The difference between the multiresolution bitmap and virtual bitmap is that for the multiresolution the strips have varying widths.

5.1.6 Adaptive sampling

Adaptive sampling is a method similar to Min-count proposed by Wegman and analyzed by Flajolet [7]. Rather than always keeping a fixed number of samples m, adaptive sampling specifies a maximum number of samples. Whenever the maximum is exceeded, half of the remaining area is cut off, and the samples that fall below the cutoff are kept.

The adaptive sampling estimator is illustrative of the difference between traditional estimators and the martingale estimator. The usual estimator for adaptive sampling is k/p where p is the remaining area in the sketch. Each item stored in the sketch equally adds $1/p$ to the estimate. For the martingale estimator, the estimator is $\sum_x \frac{1}{p_x}$ where the sum is over all points that updated the sketch, including ones not

stored in the final sketch, and p_x is the sampling probability or area at the time of adding x.

5.2 Creating new sketches: Discrete Max-count

In addition to better understanding of existing sketches, the area cutting process also leads to other sketches that may be used in approximate counting. The primary weakness of the Min-count algorithm is the number of bits required per sample stored, since each sample is a real number. We present the Discrete Max-count sketch which is a simple modification of the Min-count sketch that allows the sketch to be stored more efficiently. The Max-count sketch is obtained by hashing each distinct element to a discrete distribution F and taking the largest m values. The distribution F may be used as a tuning parameter which yields improves accuracy for targeted ranges of cardinalities. Note that when the distribution F is continuous, there is effectively no difference from the usual Min-count algorithm since F is monotonically increasing and $F(X) \sim Uniform(0,1)$ when $X \sim F$. Thus, the Min-count sketch is obtained by a simple variable transformation. We consider the case when the discrete distribution is $Geometric(1/m)$.

Under this distribution, Discrete Max-count has several interesting properties. First, it has infinite capacity since it can estimate arbitrarily large cardinalities. Second, the sketch may be decomposed into an offset which grows as $O(\log n)$ with the number of distinct elements encountered and a set of samples which does not grow with n. Third, the relative error of the algorithm is $O(1/\sqrt{m})$. Together this gives the sketch a space complexity of $O(\log n + \epsilon^{-2} g(\epsilon))$ where g is some function and ϵ is the relative error. If g is bounded by a constant, the sketch would have optimal space-complexity. Section 6.3 shows that $g(\epsilon)$ may be upper bounded by $\log \epsilon$ using a naive encoding of the samples. We also note that there is no interaction between the relative error ϵ and n in the space complexity bound. Kane et al [11] give an extensive list of space complexities for distinct counting algorithms. Only their theoretically optimal but empirically impractical [13] algorithm has this property.

We present the main tool to analyze this process, the notion of self-similarity, and give a dense set of proofs justifying these claims. An analysis of the method's running time is deferred to section 7.

6. SELF-SIMILAR AREA CUTTING

We introduce the notions of an offset decomposition and self-similarity for area cutting process. Consider a sequence of sketches S_t with areas $q(S_t)$ and jump times T_i. An area process is defined to be *self-similar* if $q(S_{T_i})/q(S_{T_{i+j}}) \stackrel{d}{=} q(S_0)/q(S_{T_j})$ for all $i > j$. In other words, it is self-similar when the proportion of the area cut out by j jumps remains the same in distribution over time. An area process is strongly self-similar if, in addition, the relative area cut off by a jump $q(S_T)/q(S_{T-1})$ is independent of $q(S_{T-1})$. If an area cutting process can be written as an offset Δ_t and a set of samples Y_t so that there is a bijection between S_t and (Δ_t, Y_t) and there exist functions q_0, q_1 such that for all t

$$q(S_t) = q_0(\Delta_t) q_1(Y_t)$$

$$Y_t \text{ is independent of } \Delta_t \text{ and } q_1(Y_t) \stackrel{d}{=} q_1(Y_1)$$

then (Δ_t, Y_t) is an offset decomposition of the area cutting process. An offset decomposition is (strongly) self-similar if the offset area $q_0(\Delta_t)$ is (strongly) self-similar and a process is a decomposable area cutting process if a non-trivial self-similar offset decomposition exists. When $0 \le q_0, q_1 \le 1$, the decomposition conditions state that the area may be obtained by first cutting off an area specified by the offset Δ_t and then independently cutting off part of the remaining area specified the samples Y_t. The condition also implies that the samples Y_t do not carry information about the cardinality since the distribution of Y_t does not depend on t. However, the samples do contain information that prevents double counting of duplicates.

The Min-count sketch is an example of a strongly self-similar area cutting process once at least m items have been encountered. The offset is equal to the m^{th} smallest item $S_t^{(m)}$, and the samples are the remaining items in the sketch scaled by $S^{(m)}$. These rescaled samples are independent $Uniform(0,1)$ and independent of the offset. The proportion of area remaining after another cut is the maximum of these samples and a new independent $Uniform(0,1)$ variable. This maximum is always $Beta(m,1)$ distributed. Hence, the remaining proportion is independent of the offset, and its distribution does not change over time.

These self-similarity conditions lead to several attractive properties. Strong self-similarity ensures that a sketch has infinite capacity if the offset cannot cut off all the area, and it allows the variance of the martingale estimator to be computed exactly. An offset decomposition ensures that the size of the sketch is well-controlled. These properties are proved in the following theorems.

THEOREM 5. *The size required to store the samples Y_t of any offset decomposition does not depend on the cardinality n. Furthermore if the remaining area $q(\Delta_t, Y_t) > 0$ for all t almost surely, then the sketch can give unbiased estimates for arbitrarily large cardinalities almost surely.*

PROOF. The size required to store the samples Y_t is purely a function of the conditional distribution $Y_t|\Delta_t$ Since Y_t is independent of Δ_t and the marginal distribution of Y_t is the same for all t, the storage requirement for Y_t does not depend on n or Δ_t. If the remaining area is always greater than 0 then the martingale estimator will give an unbiased estimate for any cardinality. □

THEOREM 6. *For any area cutting process with an offset decomposition (Δ_t, Y_t), the variance of the martingale estimator is*

$$\text{Var}(\hat{N}_n) = \sum_{t=1}^{n} \left(\mathbb{E}\left(\frac{1}{q_0(\Delta_{t-1})}\right) \mathbb{E}\left(\frac{1}{q_1(Y_1)}\right) - 1 \right) \quad (17)$$

PROOF. Consider the increments of the quadratic variation process for the martingale estimator. The expected squared increment for the t^{th} distinct item is $\frac{1}{q(S_t)^2} q(S_t)(1 - q(S_t)) = 1/q(S_t) - 1$. The rest follows from the independence of Δ_t and Y_t and the product form for the area. □

THEOREM 7. *For any strongly self-similar offset decomposition, $1/q_0(\Delta_t) - ct$ is a martingale for some constant c. The variance of the martingale estimator has an analytic form given by*

$$\text{Var}(\hat{N}_n) = n \left(\mathbb{E}\frac{1}{q_0(\Delta_0)} + c\frac{n-1}{2} \right) \mathbb{E}\frac{1}{q_1(Y_1)} - n \quad (18)$$

$$\approx \frac{n^2 r}{2} \mathbb{E}(q_1(Y_1)) \mathbb{E}(q_1(Y_1)^{-1}) \quad (19)$$

where $r = \left(\mathbb{E}\frac{q_0(\Delta_{T-1})}{q_0(\Delta_T)} - 1\right)$.

PROOF. Note that $\frac{1}{q_0(\Delta_{t+1})} = \frac{1}{q_0(\Delta_t)}\frac{q_0(\Delta_t)}{q_0(\Delta_{t+1})}$ and that the two terms on the right hand side are independent conditional on $t+1$ being a jump time. Applying the law of total probability on $t+1$ being a jump time with probability $q_0(\Delta_t)q_1(Y_t)$, it is easy to show that $1/q_0(\Delta_t) - ct$ is a martingale with $c = r\mathbb{E}q_1(Y_1)$. The variance follows from equation 17. □

This theorem shows how the two components of an offset decomposition affects the variance of the estimate. The estimates are more accurate if the expected proportion r that is cut off is small and if the expected area cut off by the sample has low variance. This may be seen from applying the delta method to obtain $\mathbb{E}(1/q_1(Y_1)) \approx 1/\mathbb{E}q_1(Y_1) + \mathrm{Var}(q_1(Y_1))/(\mathbb{E}q_1(Y_1))^2$. Furthermore, this gives a simple method to compute a closed form estimate of the error. Compute r,$\mathbb{E}(q_1(Y_1)^{-1})$, and $\mathbb{E}q_1(Y_1)$ analytically or via simulation and plug those values into equation 19.

6.1 Non-streaming estimators

Another consequence of theorem 7 is that a strongly self-similar decomposition leads to an unbiased estimator in non-streaming settings. Since $1/q_0(\Delta_t) - ct$ is a martingale,

$$\hat{N}_n^{(batch)} = \frac{1}{c}\left(\frac{1}{q_0(\Delta_n)} - \frac{1}{q_0(\Delta_0)}\right) \tag{20}$$

is an unbiased estimator that only depends on the final sketch S_n and a known initialization condition Δ_0. Note that the samples Y_n do not appear in the estimator. Since the distribution of Y_t does not depend on t and is independent of Δ_t, it is an ancillary statistic.

The variance of this estimator may be analyzed in the same manner as showing $1/q_0(\Delta_t) - ct$ is a martingale. Let $\kappa = \mathbb{E}\frac{q_0(\Delta_{T-1})^2}{q_0(\Delta_T)^2} - 1$ for a stopping time T.

$$\mathbb{E}\left(\frac{1}{q_0(\Delta_{t+1})^2}|q_0(\Delta_t)\right) = \frac{1}{q_0(\Delta_t)^2}\mathbb{E}\left(\frac{q_0(\Delta_t)^2}{q_0(\Delta_{t+1})^2}\right) \tag{21}$$

$$= \frac{1}{q_0(\Delta_t)^2} + \frac{\kappa\mathbb{E}q_1(Y_1)}{q_0(\Delta_t)} \tag{22}$$

Recursing and using the non-streaming estimator to compute $\mathbb{E}1/q_0(\Delta_t)$ yields that the variance of the non-streaming estimator is approximately

$$\mathrm{Var}(\hat{N}_n^{(batch)}) \approx \left(\frac{\kappa}{2c} - 1\right)n^2. \tag{23}$$

6.2 Streaming versus Non-streaming Min-count

For Min-count, the optimal non-streaming estimator as well as its variance are known. Since it yields a strongly self-similar area cutting process, the variance of the martingale estimator is also known. The optimal non-streaming estimator [3] is $\frac{m-1}{\Delta_t}$. Since $\Delta_t \sim Beta(m, t-m+1)$, the estimator has variance $\frac{(t-m+1)t}{m-2} \approx t^2/(m-2)$. By equation 19, the martingale estimator has asymptotic variance $\frac{t^2}{2(m-1)}$. The streaming estimator easily beats the optimal non-streaming estimator and achieves the same error with half the number of samples. Furthermore, the non-streaming estimator in equation 20 is the same as the optimal non-streaming estimator, and the variance estimate converges to the true variance as $m \to \infty$.

6.3 Analysis of Discrete Max-count

THEOREM 8. *Discrete Max-count converges almost surely to an area cutting process with a strongly self-similar decomposition.*

PROOF. Let Δ_t be the m^{th} largest element $S_t^{(m)}$ in the sketch and $Y_{ti} = S_t^{(i)} - S_t^{(m)}$. By the memoryless property of the geometric distribution, Δ_t and Y_t are independent, and this is an offset decomposition once the sketch contains at least m items. The proportion of area cut off by the offset $q(S_T)/q(S_{T-1})$ is the minimum of the samples Y_{T_i-1} and the newly observed value $h(X_T) - \Delta_{T-1}$ which is also independent of Δ_{T-1} by the memoryless property. Hence, the offset area process $q_0(\Delta_t)$ is strongly self-similar. □

The self-similarity property aids in upper bounding the variance and space-complexity of the algorithm. Since Discrete Max-count converges to a process with a strongly self-similar decomposition, the asymptotic variance is $O(r)$ where r is defined in theorem 7. The number of strips cut off by the offset at jump time T is obtained by taking the minimum of the samples Y_{T-1} along with the newly observed value $X_T - \Delta_{T-1}$. It may be upper bounded by taking the minimum of m independent $Geometric(1/m)$ variables rather than m unique $Geometric(1/m)$ variables. The probability that the minimum of m independent $Geometric(1/m)$ random variables is at most i is $1-(1-1/m)^{im}$. This is the distribution function of a $Geometric(1-(1-1/m)^m)$ variable. Using the moment generating function for the geometric distribution gives that $r < \frac{1}{m}\frac{1}{(1-(1-1/m)^m)(1-1/m)} \approx \frac{1}{m}\frac{1-e^{-1}}{\approx} 1.58/m$. The term $\gamma = \mathbb{E}q_1(Y_1)\mathbb{E}(q_1(Y_1)^{-1})$ trivially satisfied the inequality $1 < \gamma < 1/(1-1/m)^{m+1} \approx e$. Empirically, we find that $\gamma \approx 1$ for a range of $m \in [100, 10000]$. This provides an both an approximate upper bound for the variance of $1.58n^2/m$ and proves that the relative error is order $O(1/\sqrt{m})$.

The space complexity of the sketch may be analyzed by considering the size required for the offset Δ_t and for the samples Y_t. In a naive encoding of the samples, each sample is encoded as a difference from the offset, and the number of bits required for each sample is on the order of $\log(m)/\log(1-1/m) \approx m\log m = O(\epsilon^{-2}\log\epsilon)$ where ϵ is the relative error. Naively encoding the offset as an integer requires $O(m\log n)$ space due to the fine granularity of the discretization. However, the non-streaming estimator in equation 20 demonstrates that the offset is close to a function of the current estimate. The offset may be encoded as a difference from this expectation. Equations 19 and 23 establish that $\frac{1}{c}\left(\frac{1}{q_0(\Delta_n)} - \frac{1}{q_0(\Delta_0)}\right)$ have mean n and variance $O(n^2m)$. Note that $q_0(\Delta_n) = (1-1/m)^{\Delta_n}$ is smooth. A simple Chebyshev bound and Taylor expansion yields that $\Delta_0 - m\log\hat{N}_n = O_p(n\sqrt{m})$. In other words, the offset may be stored in $O_p(\log n + \log\epsilon)$ bits. This gives the space complexity of Discrete Max-count to be $O(\log n + \epsilon^{-2}\log\epsilon)$.

6.4 Analysis of Streaming HyperLogLog

A trivial decomposition for any area cutting process always exists where the samples $Y_t = \{\}$. Theorem 6 gives that the variance that is approximately the sum of expected reciprocal areas. Since HyperLogLog estimates the cardinality in terms of the reciprocal area, $\hat{N}_{hll} = \alpha_m m/q(S_t)$ where

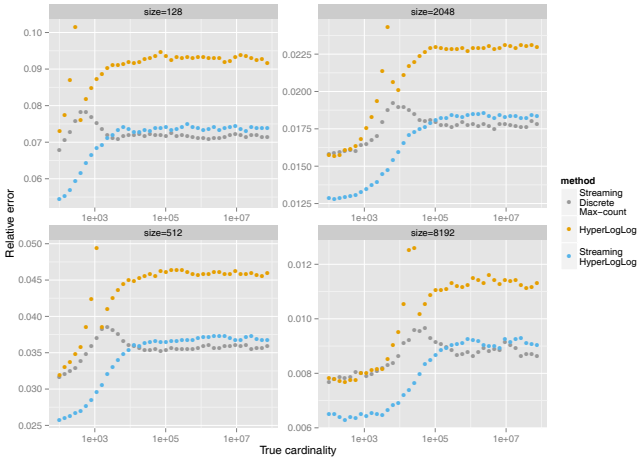

Figure 3: Relative error plot with *size* number of bins. Both streaming methods beat HyperLogLog over the entire range of cardinalities.

$\alpha_m = 0.7213/(1 + 1.079/m)$ for $m \geq 128$, substituting n for $\hat{N}_{hll}$ gives the plug-in estimate

$$\mathrm{Var}(\hat{N}_n^{(SHLL)}) \approx \frac{1}{\alpha_m m}\frac{n(n+1)}{2} - n \approx \frac{1}{1.4426\, m} n^2. \quad (24)$$

By comparison, the estimated variance for HyperLogLog is $\frac{1.04^2}{m}n^2$. In other words, HyperLogLog is predicted to use approximately $1.4426 \times 1.04^2 \approx 1.56$ times the space of Streaming HyperLogLog to achieve the same error. This matches extremely closely to our empirical evaluation where we found that Streaming HyperLogLog's variance bested HyperLogLog's by a factor of 1.57.

7. EXPERIMENTAL EVALUATION

The streaming cardinality and error estimates are empirically validated using both simulations and a real dataset. For all the simulations a 64 bit hash is used. A stream of 10^7 distinct elements is passed through each method and both cardinality and error estimates are obtained at regular intervals on a log scale. This is repeated 5×10^5 times for each method, and the results are averaged. The methods considered are HyperLogLog, Streaming HyperLogLog, and Streaming Discrete Max-count with a $Geometric(1/m)$ discretization.

To evaluate each method, we compare the relative error and effective bits per sample which are defined as

$$RelError(\hat{N}_n) = \frac{1}{n}\sqrt{\mathbb{E}(n - \hat{N}_n)^2}$$
$$m_{effective} = RelError(\hat{N}_n)^{-2}$$
$$BitPerSample(\hat{N}_n) = \frac{size_{sketch}}{m_{effective}}.$$

The definition of the effective sample size $m_{effective}$ is motivated by the fact that for many approximate counting methods including HyperLogLog, the relative error is approximately $1/\sqrt{m}$.

Note that the effective bits per sample is tied to the encoding of the sketch. For HyperLogLog, a typical implementation of the sketch which allocates five bits per bin is used. For Discrete Max-count, the sketch is encoded in three parts: the offset which is stored as an integer, a bitmap with

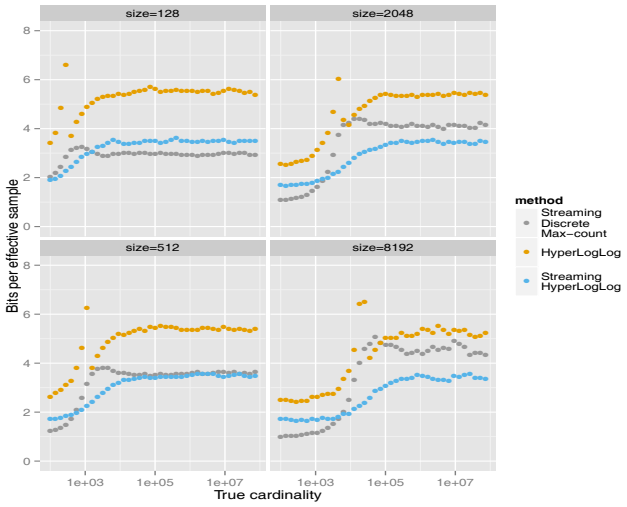

Figure 4: Effective bits per sample with *size* number of bins. For small sketches, Discrete Max-count outperforms both HyperLogLog based methods and requires half the size of HyperLogLog. For larger sketches, the encoding used by Discrete Max-count requires $O(\log m)$ bits per bin which results in worse performance for larger sketches.

$2m$ bits, and a sorted array of values above the offset plus $2m$. Each value in the array uses $\lceil \log_2 \max v - 2m \rceil$ bits where the max is taken over values in the array. We note this is not an optimal encoding of Discrete Max-Count, but it provides an upper bound on the optimal bits required per sample. In this case, the amortized running time of the method is $O_p(n \log m + m \log m \log n)$. Each distinct element requires $O(\log m)$ time to search through the sorted array, and each update requires $O(m \log m)$ time. There are $O_p(\log n / \log(1 - 1/m)) = O_p(m \log n)$ updates.

In addition to simulated data, the methods are evaluated on an anonymized Facebook test dataset. The sample contains 50 million rows containing approximately 1 million distinct users. To obtain accurate estimates of the relative error, each algorithm is run 4000 times with a randomly salted hash for each run. Figure 5 shows that the results on the dataset match the simulation.

8. DISCUSSION AND FUTURE WORK

There remain several questions of interest in this paper. In particular, questions about the space-complexity of these methods and the encoding used for each sketch remain. The ideas behind the self-similar area cutting processes and the decomposition into an offset Δ_t and set of samples Y_t leads to the Discrete Max-count sketch and space-complexities that only depend on the cardinality n through a counter. However, the encoding of the samples Y_t used in this paper has a space-complexity of $O(\epsilon^{-2} \log 1/\epsilon)$ rather than a rate of $O(\epsilon^{-2})$ to obtain the optimal space-complexity. Likewise, the FM-sketch yields a self-similar area cutting process. However, in the formulation of the process in section 6.4 the offset requires storage that grows at a rate greater than the optimal rate. We also conjecture that there are alternate decomposition of the FM-sketch into an offset and samples along with an encoding that yields optimal space-complexity. One possible offset is to take the median value of the bins.

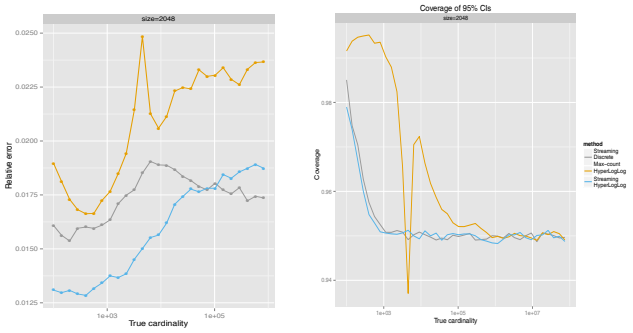

Figure 5: The left figure shows results on real data match the simulations. The right figure shows that the error estimates yield confidence intervals with the correct asymptotic coverage. HyperLogLog has worse coverage in an intermediate region due to bias. Bias corrected HyperLogLog [10] may improve on the observed coverage; however, the bias corrected method does include any improved estimates of the variance.

Another problem that remains is how to exploit the information in individual sketches that are merged. This may occur because the sketches are computed in a distributed fashion or because the cardinality of interest is the union of several sets with precomputed sketches. Although both HyperLogLog and Discrete Max-count have sketches that can be merged. Their estimators either only use the information contained in the final merged sketch or require access to the complete stream and cannot use precomputed sketches.

Another unexplored area is the effect of discretization on the underlying continuous area cutting process. HyperLogLog uses the $Geometric(1/2)$ distribution for discretization while Discrete Max-count uses the $Geometric(1-1/m)$ distribution. The discretizations lead can lead to different operating ranges as well as a different numbers of bits required per stored sample. The tradeoff between sketch size and accuracy due to discretization is not well-understood.

9. CONCLUSION

This paper presents a recipe for constructing practical algorithms that probabilistically estimate the number of distinct elements in a stream of data as well as the error of the estimate. These streaming algorithms provably outperform optimal non-streaming methods and yield substantial improvements in the storage requirements to obtain a desired relative error. This is born out in both theoretical and empirical results which show that HyperLogLog requires 50 percent more space than Streaming HyperLogLog to achieve the same error and Min-count requires twice as much space as Streaming Min-count.

We also provide a geometric interpretation to all sketches for distinct counting via the area-cutting process. Existing methods such as HyperLogLog, LPCA, and the S-bitmap are described via this process. This interpretation provides a link between different discretizations yielding different sketches and a common continuous area cutting process. Furthermore, we introduce the notion of decomposable self-similar area cutting process. Such processes have notable properties including infinite capacity and an encoding in which the sketch may be separated into two components, an offset which depends on the cardinality n and a set of samples whose the size depends only on the accuracy ϵ. This has interesting implications for the space-complexity of methods.

The proposed algorithms are analyzed under both frequentist and Bayesian settings. The cardinality estimators are proven to be optimal in various settings. In particular, the proposed martingale estimator is proven to be unbiased and optimal when queried immediately after a sketch is modified by an element in the stream. As a simple consequence of the analysis, we also give a provably optimal, unbiased estimator for the LPCA sketch which applies in both streaming and non-streaming settings.

These contributions lead to multi-faceted advances in sketch construction, cardinality and error estimation, theory, and intuition for the problem of approximate counting of distinct elements for both the streaming and non-streaming cases.

10. REFERENCES

[1] K. Aouiche and D. Lemire. A comparison of five probabilistic view-size estimation techniques in olap. In *DOLAP*, 2007.

[2] G. Casella and R. L. Berger. *Statistical inference*. Duxbury Press Belmont, CA, 2001.

[3] P. Chassaing and L. Gerin. Efficient estimation of the cardinality of large data sets. *DMTCS Proceedings*, pages 419–422, 2006.

[4] A. Chen, J. Cao, L. Shepp, and T. Nguyen. Distinct counting with a self-learning bitmap. *Journal of the American Statistical Association*, 106(495):879–890, 2011.

[5] R. Durrett. *Probability: theory and examples*, volume 3. Cambridge university press, 2010.

[6] C. Estan, G. Varghese, and M. Fisk. Bitmap algorithms for counting active flows on high speed links. In *Internet Measurement Conference*, 2003.

[7] P. Flajolet. On adaptive sampling. *Computing*, 43(4):391–400, 1990.

[8] P. Flajolet, É. Fusy, O. Gandouet, F. Meunier, et al. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *AofA*, 2007.

[9] F. Giroire. Order statistics and estimating cardinalities of massive data sets. *Discrete Applied Mathematics*, 157(2):406–427, 2009.

[10] S. Heule, M. Nunkesser, and A. Hall. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *EDBT*, 2013.

[11] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, 2010.

[12] A. Metwally, D. Agrawal, and A. E. Abbadi. Why go logarithmic if we can go linear?: Towards effective distinct counting of search traffic. In *EDBT*, 2008.

[13] J. Rae. Data stream cardinality: An empirical study of theoretically optimal algorithms. Master's thesis, University of Bristol, 2012.

[14] C. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer Texts in Statistics. Springer, 2001.

[15] K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor. A linear-time probabilistic counting algorithm for database applications. *TODS*, 1990.