LEARNING-BASED SUPPORT ESTIMATION IN SUBLINEAR TIME

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the problem of estimating the number of distinct elements in a large data set (or, equivalently, the support size of the distribution induced by the data set) from a random sample of its elements. The problem occurs in many applications, including biology, genomics, computer systems and linguistics. A line of research spanning the last decade resulted in algorithms that estimate the support up to $\pm\varepsilon n$ from a sample of size $O(\log^2(1/\varepsilon) \cdot n/\log n)$, where n is the data set size. Unfortunately, this bound is known to be tight, limiting further improvements to the complexity of this problem. In this paper we consider estimation algorithms augmented with a machine-learning-based predictor that, given any element, returns an estimation of its frequency. We show that if the predictor is correct up to a constant approximation factor, then the sample complexity can be reduced significantly, to

$$\log(1/\varepsilon) \cdot n^{1-\Theta(1/\log(1/\varepsilon))}.$$

We evaluate the proposed algorithms on a collection of data sets, using the neural-network based estimators from Hsu et al, ICLR'19 as predictors. Our experiments demonstrate substantial (up to 3x) improvements in the estimation accuracy compared to the state of the art algorithm.

1 INTRODUCTION

Estimating the support size of a distribution from random samples is a fundamental problem with applications in many domains. In biology, it is used to estimate the number of distinct species from experiments (Fisher et al., 1943); in genomics to estimate the number of distinct protein encoding regions (Zou et al., 2016); in computer systems to approximate the number of distinct blocks on a disk drive (Harnik et al., 2016), etc. The problem has also applications in linguistics, query optimization in databases, and other fields.

Because of its wide applicability, the problem has received plenty of attention in multiple fields[1], including statistics and theoretical computer science, starting with the seminal works of Good and Turing Good (1953) and Fisher et al. (1943). A more recent line of research pursued over the last decade (Raskhodnikova et al., 2009; Valiant & Valiant, 2011; 2013; Wu & Yang, 2019) focused on the following formulation of the problem: given access to independent samples from a distribution $\mathcal{P}$ over a discrete domain $\{0 \ldots n-1\}$ whose minimum non-zero mass[2] is at least $1/n$, estimate the support size of $\mathcal{P}$ up to $\pm\varepsilon n$. The state of the art estimator, due to Valiant & Valiant (2011); Wu & Yang (2019), solves this problem using only $O(n/\log n)$ samples (for a constant ε). Both papers also show that this bound is tight.

A more straightforward linear-time algorithm exists, which reports the number of distinct elements seen in a sample of size $N = O(n \log \varepsilon^{-1})$ (which is $O(n)$ for constant ε), without accounting for the unseen items. This algorithm succeeds because each element i with non-zero mass (and

[1] A partial bibliography from 2007 contains over 900 references. It is available at `https://courses.cit.cornell.edu/jab18/bibliography.html`.

[2] This constraint is naturally satisfied e.g., if the distribution $\mathcal{P}$ is an empirical distribution over a data set of n items. In fact, in this case all probabilities are multiples of $1/n$ so the support size is equal to the number of distinct elements in the data set.

thus mass at least $1/n$) appears in the sample with probability at least $1-(1-1/n)^N > 1-\varepsilon$, so in expectation, at most $\varepsilon \cdot n$ elements with non-zero mass will not appear in the sample. Thus, in general, the number of samples required by the best possible algorithm (i.e., $n/\log n$) is only logarithmically smaller than the complexity of the straightforward linear-time algorithm.

A natural approach to improve over this bound is to leverage the fact that in many applications, the input distribution is not entirely unknown. Indeed, one can often obtain rough approximations of the element frequencies by analyzing different but related distributions. For example, in genomics, frequency estimates can be obtained from the frequencies of genome regions of different species; in linguistics they can be inferred from the statistical properties of the language (e.g., long words are rare), or from a corpus of writings of a different but related author, etc. More generally, such estimates can be learned using modern machine learning techniques, given the true element frequencies in related data sets. The question then becomes whether one can utilize such predictors for use in support size estimation procedures in order to improve the estimation accuracy.

Our results In this paper we initiate the study of such learning-based methods for support size estimation. Our contributions are both theoretical and empirical. On the theory side, we show that given a good enough predictor of the distribution $\mathcal{P}$, one can solve the problem using much fewer than $n/\log n$ samples. Specifically, suppose that in the input distribution $\mathcal{P}$ the probability of element i is p_i, and that we have access to a predictor $\Pi(i)$ such that $\Pi(i) \le p_i \le b \cdot \Pi(i)$ for some constant approximation factor $b \ge 1$.[3] Then we give an algorithm that estimates the support size up to $\pm\varepsilon n$ using only

$$\log(1/\varepsilon) \cdot n^{1-\Theta(1/\log(1/\varepsilon))}$$

samples, assuming the approximation factor b is a constant (see Theorem 1 for a more detailed bound). This improves over the bound of Wu & Yang (2019) for any fixed values of the accuracy parameter ε and predictor quality factor b. Furthermore, we show that this bound is almost tight.

Our algorithm is presented in Algorithm 1. On a high level, it partitions the range of probability values into geometrically increasing intervals. We then use the predictor to assign the elements observed in the sample to these intervals, and produce a Wu-Yang-like estimate within each interval. Specifically, our estimator is based on Chebyshev polynomials (as in Valiant & Valiant (2011); Wu & Yang (2019)), but the finer partitioning into intervals allows us to use polynomials with different, carefully chosen parameters. This leads to significantly improved sample complexity if the predictor is sufficiently accurate.

On the empirical side, we evaluate the proposed algorithms on a collection of real and synthetic data sets. For the real data sets (network traffic data and AOL query log data) we use neural-network based predictors from Hsu et al. (2019). Although those predictors do not always approximate the true distribution probabilities up to a small factor, our experiments nevertheless demonstrate that the new algorithm offers substantial improvements (up to 3x reduction in relative error) in the estimation accuracy compared to the state of the art algorithm of Wu & Yang (2019).

1.1 Related work

Estimating support size As described in the introduction, the problem has been studied extensively in statistics and theoretical computer science. The best known algorithm, due to Wu & Yang (2019), uses $O(\log^2(1/\varepsilon) \cdot n/\log n)$ samples. Because of the inherent limitations of the model that uses only random samples, Canonne & Rubinfeld (2014) considered an augmented model where an algorithm has access to the *exact* probability of any sampled item. The authors show that this augmentation is very powerful, reducing the sampling complexity to only $O(1/\varepsilon^2)$. More recently, Onak & Sun (2018) proved that Canonne and Rubinfeld's algorithm works as long as the probabilities accessed are accurate up to a $(1 \pm \frac{\varepsilon}{3})$-multiplicative factor. However, this algorithm strongly relies on the probabilities being extremely accurate, and the predicted probabilities even being off by a small constant factor can cause the support size estimate to become massively incorrect. As a result, their algorithm is not robust to mispredicted probabilities, as our experiments show.

A different line of research studied *streaming algorithms* for estimating the number of distinct elements. Such algorithms have access to the whole data set, but must read it in a single pass using

[3] Our results hold without change if we modify the assumption to $r \cdot \Pi(i) \le p_i \le r \cdot b \cdot \Pi(i)$, for any $r > 0$. We use $r = 1$ for simplicity.

limited memory. The best known algorithms for this problem compute a $(1+\varepsilon)$-approximate estimation to the number of distinct elements using $O(1/\varepsilon^2 + \log n)$ bits of storage (Kane et al., 2010). See the discussion in that paper for a history of the problem and further references.

Learning-based algorithms Over the last few years, there has been a growing interest in using machine learning techniques to improve the performance of "classical" algorithms. This methodology found applications in similarity search (Wang et al., 2016; Sablayrolles et al., 2019; Dong et al., 2020), graph optimization (Khalil et al., 2017; Balcan et al., 2018), data structures (Kraska et al., 2018; Mitzenmacher, 2018), online algorithms (Lykouris & Vassilvitskii, 2018; Purohit et al., 2018), compressed sensing (Mousavi et al., 2015; Baldassarre et al., 2016; Bora et al., 2017) and streaming algorithms (Hsu et al., 2019; Jiang et al., 2019). The last two papers are closest to our work, as they solve various computational problems over data streams, including distinct elements estimation in Jiang et al. (2019) using frequency predictors. Furthermore, in our experiments we are using the neural-network-based predictors developed in Hsu et al. (2019). However, our algorithm operates in a fundamentally different model, using a sublinear (in n) number of samples of the input, as opposed to accessing the full input via a linear scan. Thus, our algorithms run in *sublinear time*, in contrast to streaming algorithms that use *sublinear space*.

Distribution property testing This work can be seen more broadly in the context of testing properties of distributions over large discrete domains. Such questions are studied at the crossroads of social networks, statistics, information theory, database algorithms, and machine learning algorithms. Examples of specific properties that have been extensively considered include testing whether the distribution is uniform, Gaussian, high entropy, independent or monotone increasing (see e.g. Rubinfeld (2012); Canonne (2015); Goldreich (2017) for surveys on the topic).

2 Learning-Based Algorithm

2.1 Preliminaries

Problem setting and notation. The support estimation problem is formally defined as follows. We are given sample access to an unknown distribution $\mathcal{P}$ over a discrete domain of size n. For simplicity, we identify the domain with $[n] = \{1, \ldots, n\}$. Let p_i denote the probability of element i. Let $\mathcal{S}(\mathcal{P}) = \{i : p_i > 0\}$ be the support of $\mathcal{P}$. Our goal is to estimate the support size $S = |\mathcal{S}(\mathcal{P})|$ using as few samples as possible. In particular, given $\varepsilon > 0$, the goal is to output an estimate $\tilde{S}$ that satisfies $\tilde{S} \in [S - \varepsilon n, S + \varepsilon n]$.

We assume that the minimal non-zero mass of any element is at least $1/n$, namely, that $p_i \geq 1/n$ for every $i \in \mathcal{S}(\mathcal{P})$. This is a standard promise in the support estimation problem (see, e.g., Raskhodnikova et al. (2009); Valiant & Valiant (2011); Wu & Yang (2019)), and as mentioned earlier, it naturally holds in the context of counting distinct elements, where p_i is defined as the count of element i in the sample divided by n. Furthermore, a lower bound on the minimum non-zero probability is a necessary assumption without which no estimation algorithm is possible, even if the number of samples is allowed to be an arbitrarily large function of n, i.e., not just sublinear algorithms. The reason is that there could be arbitrarily many elements with exceedingly small probabilities that would never be observed. See for example the discussion in the supplementary Section 5 of Orlitsky et al. (2016).

In the learning-based setting, we furthermore assume we have a predictor Π that can provide information about p_i. In our analysis, we will assume that $\Pi(i)$ is a constant factor approximation of each p_i. In order to bound the running time of our algorithms, we assume we are given access to a ready-made predictor and (as in Canonne & Rubinfeld (2014)) that evaluating $\Pi(i)$ takes unit time. In our experiments, we use neural network based predictors from Hsu et al. (2019). In general, predictors need to be trained (or otherwise produced) in advance. This happens in a preprocessing stage, before the input distribution is given to the algorithm, and this stage is not accounted for in the sublinear running time. We also note that training the predictor needs to be done only once for all future inputs (not once per input).

The Wu & Yang (2019) estimator. In the classical setting (without access to a predictor), Wu & Yang (2019) gave a sample-optimal algorithm based on Chebyshev polynomials. We now describe it briefly, as it forms the basis for our learning-based algorithm.

Suppose we draw N samples, and let N_i be the number of times element i is observed. The output estimate of Wu & Yang (2019) is of the form

$$\tilde{S}_{\mathrm{WY}} = \sum_{i \in [n]} (1 + f(N_i)),$$

where $f(N_i)$ is a correction term intended to compensate for the fact that some elements in the support do not appear in the sample at all. If $p_i = 0$, then necessarily $N_i = 0$ (as i cannot appear in the sample). Thus, choosing $f(0) = -1$ ensures that unsupported elements contribute nothing to $\tilde{S}_{\mathrm{WY}}$. On the other hand, if $p_i > \frac{\log n}{N}$, then by standard concentration we have $N_i > \Omega(\log n)$ with high probability; thus choosing $f(N_i) = 0$ for all $N_i > L = \Omega(\log n)$ ensures that high-mass elements are only counted once in $\tilde{S}_{\mathrm{WY}}$. It remains to take care of elements i with $p_i \in [\frac{1}{n}, \frac{\log n}{N}]$.

By a standard Poissonization trick, the expected additive error $|S - \mathbb{E}[\tilde{S}_{\mathrm{WY}}]|$ can be bounded by $\sum_{i \in [n]} |P_L(p_i)|$, where P_L is the degree-L polynomial

$$P_L(x) = \sum_{k=0}^{L} \frac{\mathbb{E}[N]^k}{k!} \cdot f(k) \cdot x^k.$$

To make the error as small as possible, we would like to choose $f(1), \ldots, f(L)$ so as to minimize $|P_L(p_i)|$ on the interval $p_i \in [\frac{1}{n}, \frac{\log n}{N}]$, under the constraint $P_L(0) = -1$ (which is equivalent to $f(0) = -1$). This is a well-known extremal problem, and its solution is given by Chebyshev polynomials, whose coefficients have a known explicit formula. Indeed, Wu & Yang (2019) show that choosing $f(1), \ldots, f(L)$ such that $\frac{\mathbb{E}[N]^k}{k!} f(k)$ are the coefficients of an (appropriately shifted and scaled) Chebyshev polynomial leads to an optimal sample complexity of $O(\log^2(1/\varepsilon) \cdot n / \log n)$.

2.2 Our Algorithm

Algorithm 1: Learning-Based Support Estimation

Input: Number of samples N, domain size n, base b, polynomial degree L, predictor Π

Output: Estimate $\tilde{S}$ of the support size

Partition $[\frac{1}{n}, 1]$ into intervals $I_j = \left[\frac{b^j}{n}, \frac{b^{j+1}}{n}\right]$ for $j = 0, \ldots, \log_b n$

Let $a_0, a_1, \ldots, a_L$ be the coefficient of the Chebyshev polynomial $P_L(x)$ from Equation (1) (Note that $a_k = 0$ for all $k > L$)

Draw N random samples

$N_i = \#$ of times we see element i in samples

for *every interval* I_j **do**

  if $\frac{b^j}{n} \leq \frac{0.5 \log n}{N}$ **then**

    $\tilde{S}_j = \sum_{i \in [n] : \Pi(i) \in I_j} \left(1 + a_{N_i} \left(\frac{n}{b^j}\right)^{N_i} \cdot \frac{N_i!}{N^{N_i}}\right)$

  else

    $\tilde{S}_j = \#\{i \in [n] : N_i \geq 1, \Pi(i) \in I_j\}$

  end

end

return $\tilde{S} = \sum_{j=0}^{\log_b n} \tilde{S}_j$

Our main result is a sample-optimal algorithm for estimating the support size of an unknown distribution, where we are given access to samples as well as approximations to the probabilities of the elements we sample. Our algorithm is presented in Algorithm 1. It partitions the interval $[\frac{1}{n}, \frac{\log n}{N}]$

into geometrically increasing intervals $\{[\frac{b^j}{n}, \frac{b^{j+1}}{n}] : j = 0, 1, \ldots\}$, where b is a fixed constant that we refer to as the *base* parameter (in our proofs this parameter upper bounds the approximation factor of the predictor, which is why we use the same letter to denote both; its setting in practice is studied in detail in the next section). The predictor assigns the elements observed in the sample to intervals, and the algorithm computes a Chebyshev polynomial estimate within each interval. Since the approximation quality of Chebyshev polynomials is governed by the ratio between the interval endpoints (as well as the polynomial degree), this assignment can be leveraged to get more accurate estimates, improving the overall sample complexity. Our main theoretical result is the following.

Theorem 1. *Let $b > 1$ be a fixed constant. Suppose we have a predictor that given $i \in [n]$ sampled from the input distribution, outputs $\Pi(i)$ such that $\Pi(i) \leq p_i \leq b \cdot \Pi(i)$. Then, for any $\varepsilon > n^{-1/2}$, if we set $L = O(\log(1/\varepsilon))$ and draw $N \sim Poisson\left(L \cdot n^{1-1/L}\right)$ samples, Algorithm 1 reports an estimate $\tilde{S}$ that satisfies $\tilde{S} \in [S - \varepsilon\sqrt{nS}, S + \varepsilon\sqrt{nS}]$ with probability at least $9/10$. In other words, using*

$$O\left(\log(1/\varepsilon) \cdot n^{1-\Theta(1/\log(1/\varepsilon))}\right)$$

samples, we can approximate the support size S up to an additive error of $\varepsilon\sqrt{nS}$, with probability at least $9/10$.

Note that sample complexity of Wu & Yang (2019) (which is optimal without access to a predictor) is nearly linear in n, while Theorem 1 gives a bound that is polynomially smaller than n for every fixed $\varepsilon > 0$. Also, note that $\sqrt{nS} \leq n$, so our estimate $\tilde{S}$ is also within $\varepsilon \cdot n$ of the support size S.

We also prove a corresponding lower bound, proving that the above theorem is essentially tight.

Theorem 2. *Suppose we have access to a predictor that returns $\Pi(i)$ such that $\Pi(i) \leq p_i \leq 2 \cdot \Pi(i)$ for all sampled i. Then any algorithm that estimates the support size up to an εn additive error with probability at least $9/10$ needs $\Omega(n^{1-\Theta(1/\log(1/\varepsilon))})$ samples.*

We prove Theorems 1 and 2 in Appendix A. We note that while our upper bound proof follows a similar approach to Wu & Yang (2019), our lower bound follows a combinatorial approach differing from their linear programming arguments.

The Chebyshev polynomial. For completeness, we explicitly write the polynomial coefficients used by Algorithm 1. The standard Chebyshev polynomial of degree L on the interval $[-1, 1]$ is given by[4]

$$Q_L(x) = \cos(L \cdot \arccos(x)).$$

For Algorithm 1, we want a polynomial as follows:

$$P_L(x) = \sum_{k=0}^{L} a_k x^k \text{ satisfying } P_L(0) = -1 \text{ and } P_L(x) \in [-\varepsilon, \varepsilon] \text{ for all } 1 \leq x \leq b^2.$$

This is achieved by shifting and scaling Q_L, namely, this polynomial can be written as

$$P_L(x) = -\frac{Q_L\left(\frac{2x-(b^2+1)}{(b^2-1)}\right)}{Q_L\left(-\frac{b^2+1}{b^2-1}\right)}, \tag{1}$$

where ε equals $\left|Q_L\left(-\frac{b^2+1}{b^2-1}\right)\right|^{-1}$, which decays as $e^{-\Theta(L)}$ if b is a constant. Thus it suffices to choose $L = O(\log(1/\varepsilon))$ in Theorem 1.

3 Experiments

In this section we evaluate our algorithm empirically on real and synthetic data.

[4]The fact this is indeed a polynomial is proven in standard textbooks, e.g., Timan et al. (1963).

Table 1: Datasets used in our experiments. The listed values of n (total size) and support size (distinct elements) for AOL/IP are per day/minute (respectively), approximated across all days/minutes.

Name	Type	# Distributions	Predictor	n	Support size
AOL	Keywords	90 (days)	RNN	$\sim 4 \cdot 10^5$	$\sim 2 \cdot 10^5$
IP	IP addresses	60 (minutes)	RNN	$\sim 3 \cdot 10^7$	$\sim 10^6$
Zipfian	Synthetic	1	Empirical	$\sim 2 \cdot 10^5$	10^5

Datasets. We use two real and one synthetic datasets:

- **AOL**: 21 million search queries from 650 thousand users over 90 days. The queries are keywords for the AOL search engine. Each day is treated as a separate input distribution. The goal is to estimate the number of distinct keywords.
- **IP**: Packets collected at a backbone link of a Tier1 ISP between Chicago and Seattle in 2016 over 60 minutes.[5] Each packet is annotated with the sender IP address, and the goal is to estimate the number of distinct addresses. Each minute is treated as a separate distribution.
- **Zipfian**: Synthetic dataset of samples drawn from a finite Zipfian distribution over $\{1, \ldots, 10^5\}$ with the probability of each element i proportional to $i^{-0.5}$.

The AOL and IP datasets were used in Hsu et al. (2019), who trained a recurrent neural network (RNN) to predict the frequency of a given element for each of those datasets. We use their trained RNNs as predictors. For the Zipfian dataset we use the empirical counts of an independent sample as predictions. A more detailed account of each predictor is given later in this section. The properties of the datasets are summarized in Table 1.

Baselines. We compare Algorithm 1 with two existing baselines:

- The algorithm of Wu & Yang (2019), which is the state of the art for algorithms without predictor access. We abbreviate its name as WY.[6]
- The algorithm of Canonne & Rubinfeld (2014), which is the state of the art for algorithms with access to a *perfect* predictor. We abbreviate its name as CR.

Error measurement. We measure accuracy in terms of the relative error $|1 - \tilde{S}/S|$, where S is the true support size and $\tilde{S}$ is the estimate returned by the algorithm. We report median errors over 50 independent executions of each experiment, $\pm$ one standard deviation.

Summary of results. Our experiments show that on one hand, our algorithm can indeed leverage the predictor to get significantly improved accuracy compared to WY. On the other hand, our algorithm is robust to different predictors: while the CR algorithm performs extremely well on one dataset (AOL), it performs poorly on the other two (IP and Zipfian), whereas our algorithm is able to leverage the predictors in those cases too and obtain significant improvement over both baselines.

3.1 Base Parameter Selection

Algorithm 1 uses two parameters that need to be set: The polynomial degree L, and the base parameter b. The performance of the algorithm is not very sensitive to L, and for simplicity we use the same setting as Wu & Yang (2019), $L = \lfloor 0.45 \log n \rfloor$. The setting of b requires more care.

Recall that b is a constant used as the ratio between the maximum and minimum endpoint of each interval I_j in Algorithm 1. There are two reasons why b cannot be chosen too small. One is that the algorithm is designed to accommodate a predictor that provides a b-approximation of the true

[5] From CAIDA internet traces 2016, `https://www.caida.org/data/monitors/passive-equinix-chicago.xml`.

[6] Apart from their tight analysis, Wu & Yang (2019) also report experimental results showing their algorithm is empirically superior to previous baselines.

probabilities, so larger b makes the algorithm more robust to imperfect predictors. The other reason is that small b means using many small intervals, and thus a smaller number of samples assigned to each interval. This leads to higher noise in the Chebyshev polynomial estimators invoked within each interval (even if the assignment of elements to intervals is correct), and empirically impedes performance. On the other hand, if we set b to be too large (resulting in one large interval the covers almost the whole range of p_i's), we are essentially not using information from the predictor, and thus do not expect to improve over WY.

To resolve this issue, we introduce a sanity check in each interval I_j, whose goal is to rule out bases that are too small. The sanity check passes if $\tilde{S}_j \in [0, 1/l_j]$, where l_j is the left endpoint of I_j (i.e., $I_j = [l_j, r_j]$), and $\tilde{S}_j$ is as defined in Algorithm 1. The reasoning is as follows. On one hand, the Chebyshev polynomial estimator which we use to compute $\tilde{S}_j$ can in fact return negative numbers, leading to failure modes with $\tilde{S}_j < 0$. On the other hand, since all element probabilities in I_j are lower bounded by l_j, it can contain at most $1/l_j$ elements. Therefore, any estimate $\tilde{S}_j$ of the number of elements in I_j which is outside $[0, 1/l_j]$ is obviously incorrect.

In our implementation, we start by running Algorithm 1 with $b = 2$. If any interval fails the sanity check, we increment b and repeat the algorithm (with the same set of samples). The final base we use is twice the minimal one such that all intervals pass the sanity check, where the final doubling is to ensure we are indeed past the point where all checks succeed.

The effect of this base selection procedure on each of our datasets is depicted in Figures 1, 3, and 5.[7] For a fixed sample size, we plot the performance of our algorithm (dotted blue) as the base increases compared to WY (solid orange, independent of base), as well as the fraction of intervals that failed the sanity check (dashed green). The plots show that for very small bases, the sanity check fails on some intervals, and the error is large. When the base is sufficiently large, all intervals pass the sanity check, and we see a sudden plunge in the error. Then, as the base continues to grow, our algorithm continues to perform well, but gradually degrades and converges to WY due to having a single dominating interval. This affirms the reasoning above and justifies our base selection procedure.

3.2 Results

AOL data. As the predictor, we use the RNN trained by Hsu et al. (2019) to predict the frequency of a given keyword. Their predictor is trained on the first 5 days and the 6th day is used for validation. The results for day #10 are shown in Figure 2. (The results across all days are similar; see more below.) They show that our algorithm performs significantly better than WY. Nonetheless, CR (which relies on access to a perfect predictor) achieves better performance on a much smaller sample size. This is apparently due to highly specific traits of the predictor; as we will see presently, the performance of CR is considerably degraded on the other datasets.

IP data. Here too we use the trained RNN of Hsu et al. (2019). It is trained on the first 7 minutes and the 8th minute is used for validation. However, unlike AOL, Hsu et al. (2019) trained the RNN to predict the *log* of the frequency of the given IP address, rather than the frequency itself, due to training stability considerations. To use it as a predictor, we exponentiate the RNN output. This inevitably leads to less accurate predictions.

The results for minute 59 are shown in. Figure 4. (As in the AOL data, the results across all minutes are similar; see more below). Again we see a significant advantage to our algorithm over WY for small sample sizes. Here, unlike the AOL dataset, CR does not produce good results.

Zipfian data. To form a predictor for this synthetic distribution, we drew a random sample of size 10% of n, and used the empirical count of each element in this fixed sample as the prediction for its frequency. If the predictor is queried for an element that did not appear in the sample, its predicted probability is reported as the minimum $1/n$. We use this fixed predictor in all repetitions of the experiment (which were run on fresh independent samples). The results are reported in Figure 6.

[7] In those plots, for visualization, in order to compute the error whenever a sanity check fails in I_j, we replace $\tilde{S}_j$ with a naïve estimate, defined as the number of distinct sample elements assigned to I_j. Note that our implementation never actually uses those naïve estimates, since it only uses bases that pass all sanity checks.

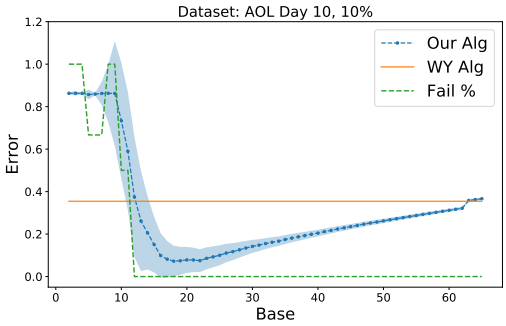


Figure 1: Error by base, AOL, sample size $10\% \cdot n$

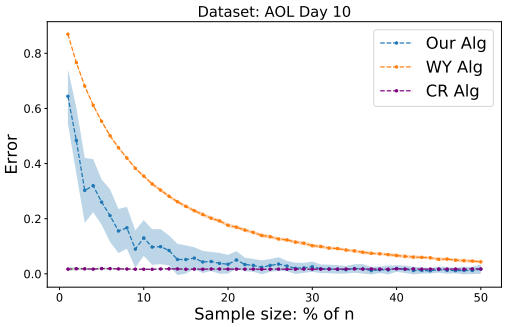


Figure 2: Error per sample size, AOL

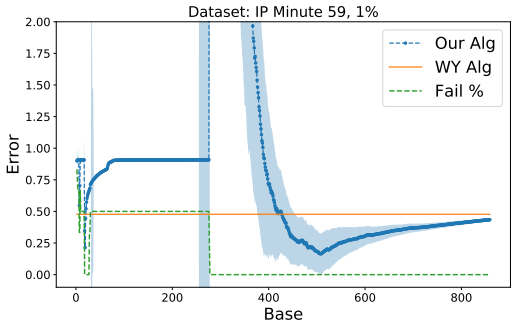


Figure 3: Error by base, IP, sample size $1\% \cdot n$

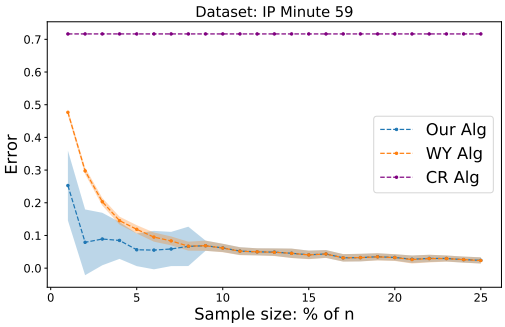


Figure 4: Error per sample size, IP

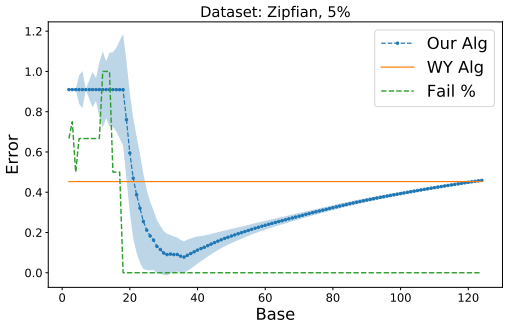


Figure 5: Error by base, Zipfian, sample size $5\%n$

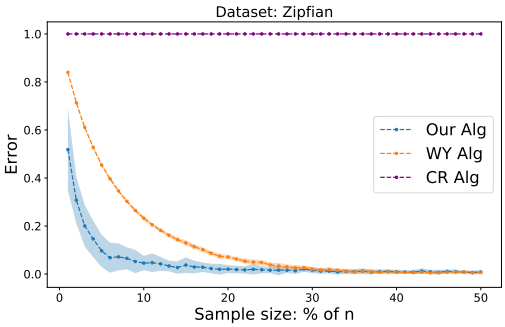
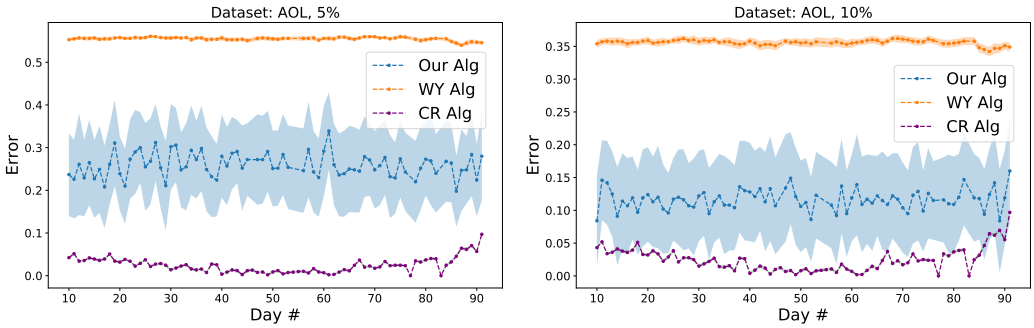


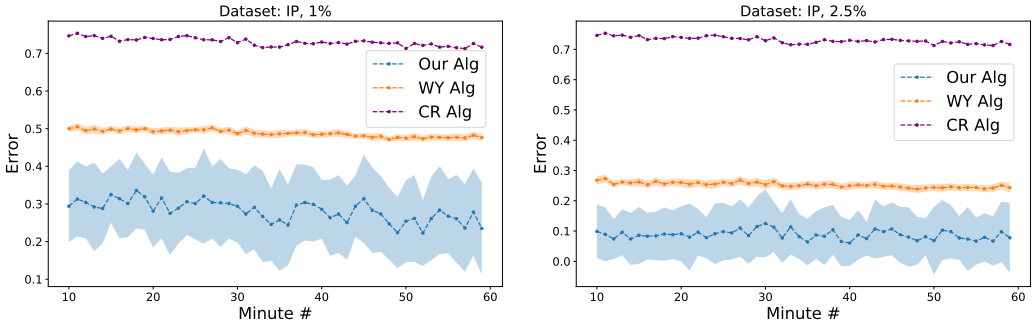
Figure 6: Error per sample size, Zipfian



(a) Sample size: $5\% \cdot n$

(b) Sample size: $10\% \cdot n$

Figure 7: Error across the different AOL days



(a) Sample size: $1\% \cdot n$

(b) Sample size: $2.5\% \cdot n$

Figure 8: Error across the different IP minutes

As with the IP data, our algorithm significantly improves over WY for small sample sizes, and both algorithms outperform CR by a large margin.

AOL and IP results over time. Finally, we present accuracy results over the days/minutes of the AOL/IP datasets (respectively). The purpose is to demonstrate that the performance of our algorithm remains consistent over time, even when the data has moved away from the initial training period and the predictor may become 'stale'. The results for AOL are shown in Figures 7a, 7b, yielding a median **2.2-fold** and **3.0-fold** improvement over WY (for sample sizes $5\% \cdot n$ and $10\% \cdot n$, respectively). The results for IP are shown in Figures 8a and 8b, yielding a median **1.7-fold** and **3.0-fold** improvement over WY (for sample sizes $1\% \cdot n$ and $2.5\% \cdot n$, respectively). As before, CR performs better than either algorithm on AOL, but fails by a large margin on IP.

REFERENCES

Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International Conference on Machine Learning*, pp. 353–362, 2018.

Luca Baldassarre, Yen-Huan Li, Jonathan Scarlett, Baran Gözcü, Ilija Bogunovic, and Volkan Cevher. Learning-based compressive subsampling. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):809–822, 2016.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pp. 537–546, 2017.

Clément Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *International Colloquium on Automata, Languages, and Programming*, pp. 283–295. Springer, 2014.

Clément L. Canonne. A survey on distribution testing: Your data is big. but is it blue? *Electron. Colloquium Comput. Complex.*, 22:63, 2015.

Yihe Dong, P. Indyk, Ilya P. Razenshteyn, and T. Wagner. Learning space partitions for nearest neighbor search. In *ICLR*, 2020.

Ronald A Fisher, A Steven Corbet, and Carrington B Williams. The relation between the number of species and the number of individuals in a random sample of an animal population. *The Journal of Animal Ecology*, pp. 42–58, 1943.

Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

Irving J Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

Danny Harnik, Ety Khaitzin, and Dmitry Sotnikov. Estimating unseen deduplicationfrom theory to practice. In *14th {USENIX} Conference on File and Storage Technologies ({FAST} 16)*, pp. 277–290, 2016.

Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019.

Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P Woodruff. Learning-augmented data stream algorithms. In *International Conference on Learning Representations*, 2019.

Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 41–52, 2010.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pp. 6348–6358, 2017.

Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 489–504, 2018.

Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *International Conference on Machine Learning*, pp. 3302–3311, 2018.

V. A. Markov. On functions of least deviation from zero in a given interval. *St. Petersburg*, 1892.

Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems*, pp. 464–473, 2018.

Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pp. 1336–1343. IEEE, 2015.

Krzysztof Onak and Xiaorui Sun. Probabilityrevealing samples. In *International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 2018–2026. PMLR, 2018.

Alon Orlitsky, Ananda Theertha Suresh, and Yihong Wu. Optimal prediction of the number of unseen species. *Proceedings of the National Academy of Sciences*, 113(47):13283–13288, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1607774113. URL `https://www.pnas.org/content/113/47/13283`.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, pp. 9661–9670, 2018.

Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009.

Ronitt Rubinfeld. Taming big probability distributions. *XRDS*, 19(1):24–28, 2012.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Herv Jgou. Spreading vectors for similarity search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkGuG2R5tm.

AF Timan, M Stark, IN Sneddon, and S Ulam. Theory of approximation of functions of a real variable. 1963.

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 forty-third annual ACM symposium on Theory of computing*, pp. 685–694, 2011.

Paul Valiant and Gregory Valiant. Estimating the unseen: improved estimators for entropy and other properties. In *Advances in Neural Information Processing Systems*, pp. 2157–2165, 2013.

Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data - a survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.

Yihong Wu and Pengkun Yang. Chebyshev polynomials, moment matching, and optimal estimation of the unseen. *The Annals of Statistics*, 47(2):857–883, 2019.

James Zou, Gregory Valiant, Paul Valiant, Konrad Karczewski, Siu On Chan, Kaitlin Samocha, Monkol Lek, Shamil Sunyaev, Mark Daly, and Daniel G MacArthur. Quantifying unobserved protein-coding variants in human populations provides a roadmap for large-scale sequencing projects. *Nature communications*, 7(1):1–5, 2016.