LEARNING-BASED SUPPORT ESTIMATION IN SUBLINEAR TIME

Talya Eden, MIT — Piotr Indyk, MIT — Shyam Narayanan, MIT — Ronitt Rubinfeld, MIT — Sandeep Silwal, MIT — Tal Wagner, MSR

Code available at:
https://github.com/ssilwa/Learning-augmented-support-estimation

Introduction

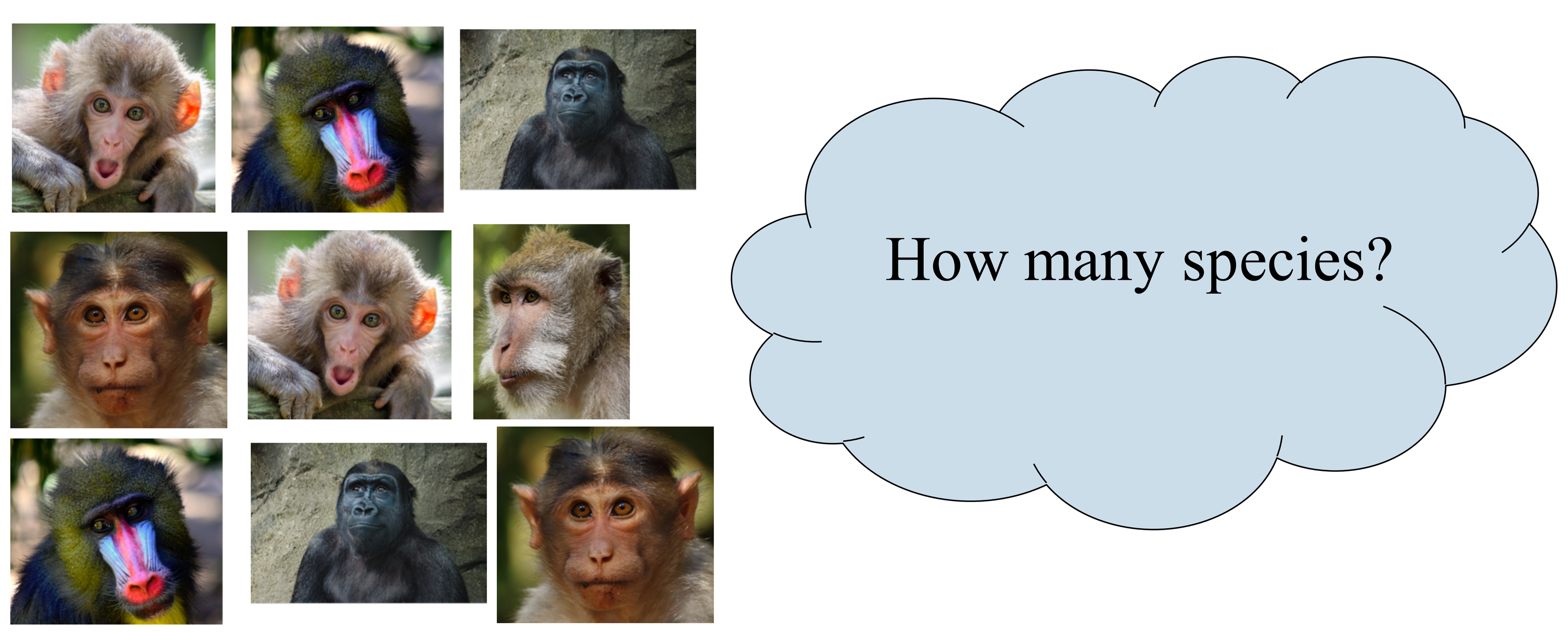
Setting: Sample access to an unknown distribution $\mathcal{P} = (p_1, \ldots, p_n)$ over domain $[n] = \{1, \cdots, n\}$

Goal: Estimate the support size $S = |i: p_i > 0|$ up to $\pm \epsilon n$ using **few** samples

- Example of **distinct elements** if $p_i = \frac{count\ of\ element\ i}{n}$
- Applications in search engines (How many distinct queries?) Biology (How may distinct species?) etc

Promise: For every i, either $p_i \geq 1/n$ or $p_i = 0$

- Naturally holds in distinct element setting



Contribution

Our Algorithm

- **Learning Based:** Assume predictor that $\prod$ such that
$$\Pi(i) \leq p_i \leq C \cdot \Pi(i)$$
for every sample i for constant $C > 0$
- Empirically use **ML driven** predictors (RNN)
- **Sublinear** sample complexity: n^c for $c < 1$
- Experimentally **robust** to noisy predictors ("sanity check" to fall back on previous best algorithm)

References

- Clement Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregateddata. InInternational Colloquium on Automata, Languages, and Programming, pp. 283–295.Springer, 2014
- Yihong Wu and Pengkun Yang. Chebyshev polynomials, moment matching, and optimal estimationof the unseen.The Annals of Statistics, 47(2):857–883, 2019.

Learning-based Algorithm

Estimator: $\sum_i \left(1 + h(N_i)\right) = S + \sum_i h(N_i)$

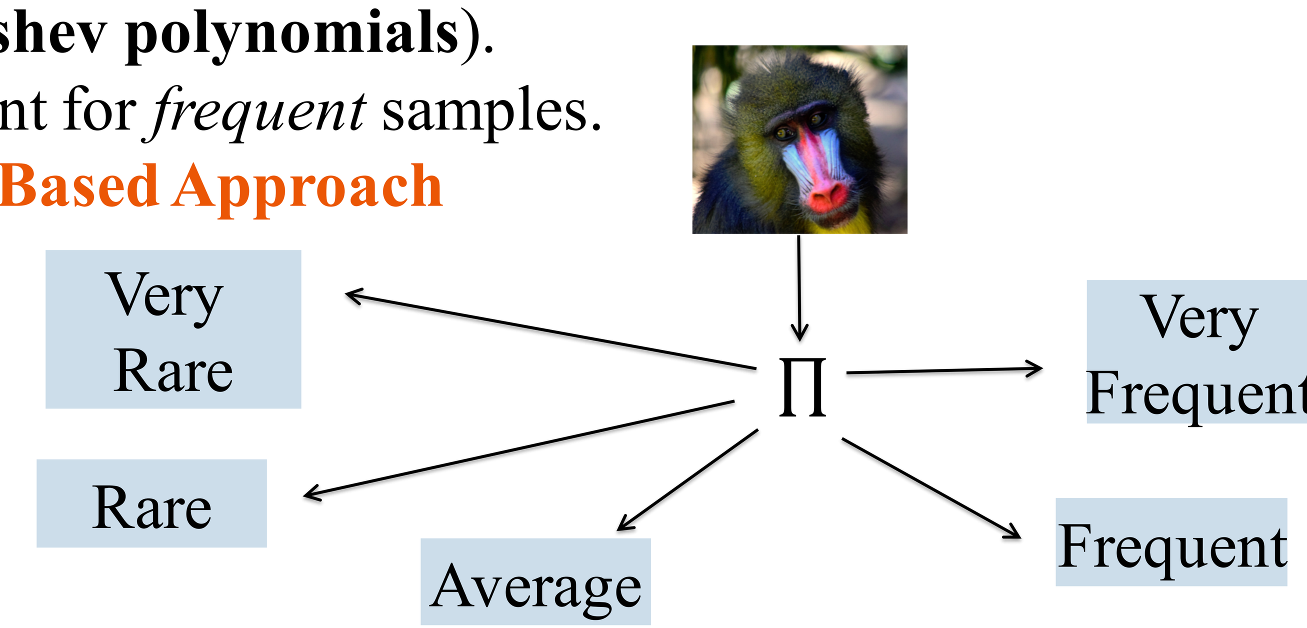
- for $h: \{0\} \cup \mathbb{N} \to \mathbb{R}$ with $h(0) = -1$
- N_i = # of samples of i th element.

Bias: $\mathbb{E}[Est - S] = \sum_i \mathbb{E}[h(N_i)]$

Previous Algorithms:

- Pick "best" h to minimize bias for *rare* samples (**Chebyshev polynomials**).
- Just count for *frequent* samples.

Learning-Based Approach



Tailor polynomial estimator for each "bucket"

Theoretical Results

Parameters: n = domain size, ϵ = error, $L = O(\log \epsilon^{-1})$

Reference	# Samples	Predictor Model
(Wu, Yang '19)	$\Theta\left(\frac{n}{\log n} \log^2\left(\frac{1}{\epsilon}\right)\right)$	No predictor
(Clemont, Rubinfeld '13)	$\Theta\left(\frac{1}{\epsilon^2}\right)$	Perfect predictor
This Work	$\Theta(Ln^{1-1/L})$	Imperfect predictor

Optimal Samples: **Any** algorithm with constant factor predictor **must** use $\Omega(Ln^{1-1/L})$ samples

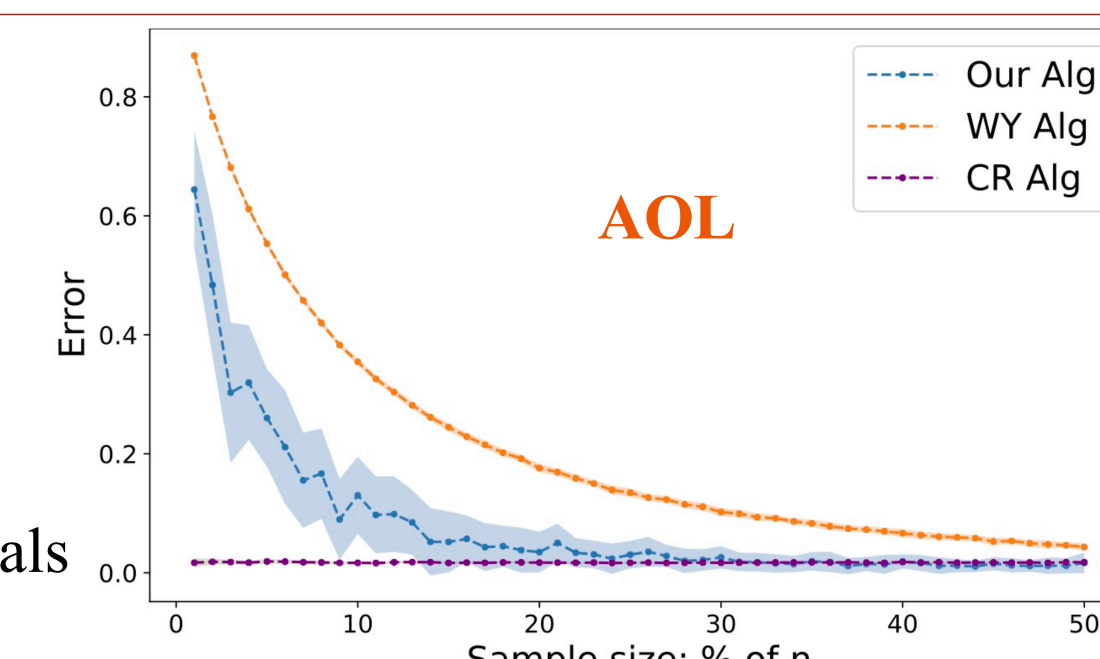
Natural Predictor Model: Cannot replace predictor with models such as additive error Π is close to $\mathcal{P}$ in TV distance or additive approximations

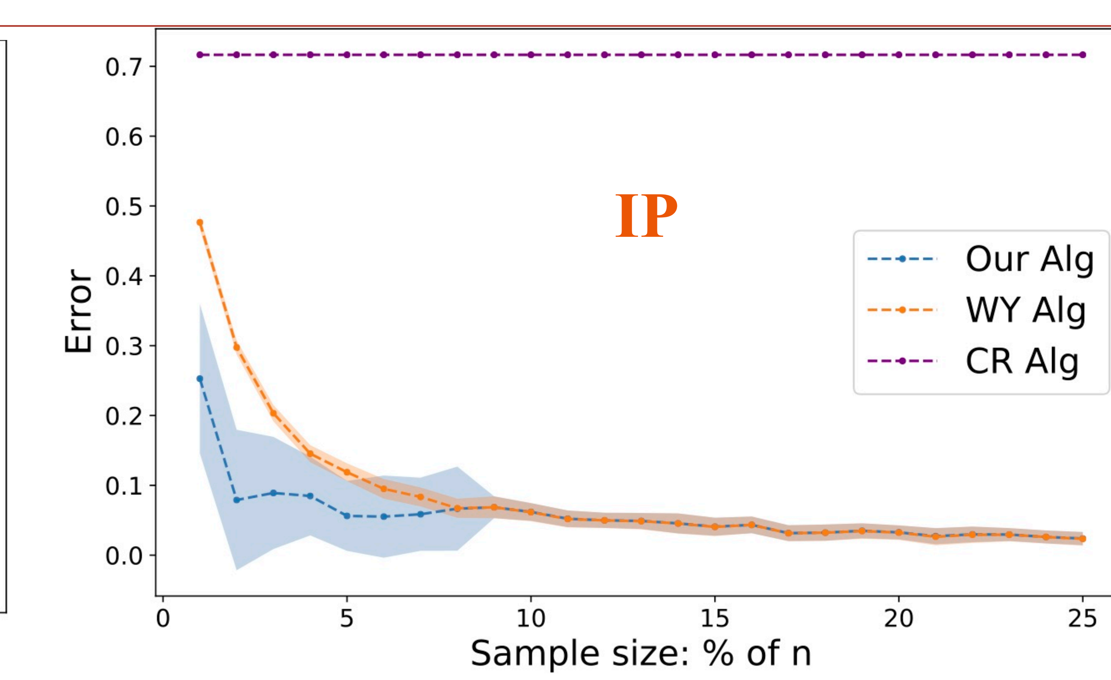
Experiments

Benchmarks:

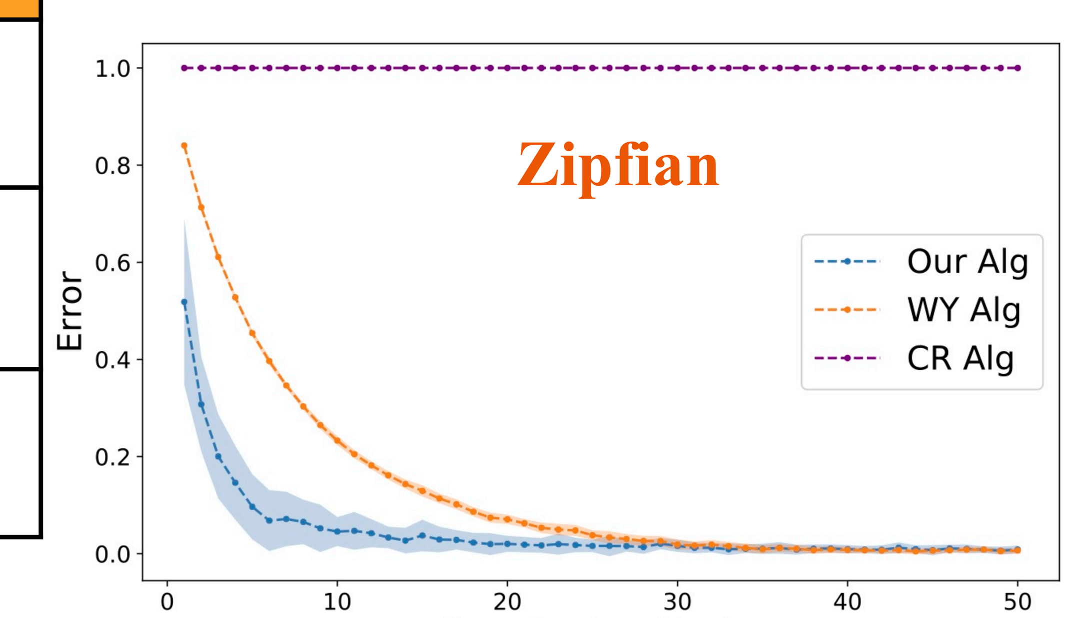
- **WY:** (Wu, Yang '19) Optimal for no predictors
- **CR:** (Clemont, Rubinfeld '13) Optimal for perfect predictors

Error: Report $|1 - Est/S|$ Averaged over 50 independent trials





Dataset	n	Predictor
AOL (Search queries)	$\sim 4 \cdot 10^5$	RNN (Hsu et al. '19)
IP (IP addresses)	$\sim 3 \cdot 10^7$	RNN (Hsu et al. '19)
Zipfian (Synthetic)	$\sim 2 \cdot 10^5$	Empirical count



Conclusion:

- Predictors can be leveraged to outperform previously optimal algorithms (**WY**)
- **CR** can fail badly sometimes whereas we are robust against different predictors
- Predictors can still be useful for data far in the future (not shown)